Universidade de Aveiro Departamento de Engenharia Mecânica
2014

Nuno Miguel Rego Araújo

Sistemas de suporte à Integração da Produção

Universidade de Aveiro Departamento de Engenharia Mecânica
2014

Nuno Miguel Rego Araújo

Sistemas de suporte à Integração da Produção

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Ana Maria Pinto de Moura, Professora Auxiliar do Departamento de Economia, Gestão e Engenharia Industrial da Universidade de Aveiro

O júri / The jury

Presidente / President

Prof. Doutor Fernando José Neto da Silva
Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro

Vogais / Committee

Prof. Doutor José Paulo Oliveira Santos
Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro (orientador)

Prof. Doutor Bernardo Sobrinho Simões de Almada Lobo
Professor Associado do Departamento de Engenharia e Gestão Industrial da Faculdade de Engenharia da Universidade do Porto

Agradecimentos / Acknowledgements

Agradeço ao meu orientador Professor José Paulo Santos e à minha co-orientadora Professora Ana Moura pelo apoio, ajuda e disponibilidade para levar este trabalho a bom termo.

Agradeço ao Engenheiro Arménio da GNS pela disponibilidade e recetividade com que sempre me acolheu.

Agradeço ao Engenheiro Hugo Antunes da empresa GNS, pela constante disponibilidade mostrada e apoio prestado no decorrer deste trabalho.

Agradeço à minha namorada Ana Vitorino por todo o apoio, carinho, motivação e incentivo transmitidos no decorrer deste trabalho.

Aos meus pais que sempre me apoiaram e proporcionaram as condições necessárias à minha progressão.

Agradeço a todos aqueles que de alguma forma contribuíram para este trabalho.

A todos muito obrigado!

Palavras-chave Serviços *Web*; Arquitetura Orientada a Serviços (SOA); Flexibilidade; Integração; Sequenciamento da produção, Algoritmo Genético

Resumo

As organizações modernas são caracterizadas pela procura de maior flexibilidade e interoperabilidade. Dentro das organização podem ser encontrados um grande número de sistemas desde aplicações informáticas que cobrem todo o nível superior aos dispositivos distribuídos dos níveis mais inferiores do chão da fábrica. Ao mesmo tempo as organizações também procuram uma maior integração dos seus sistemas de informação com os dispositivos e equipamentos dos níveis inferiores. A integração destes sistemas veio trazer novos obstáculos devido às diferenças na forma como todos os elementos comunicam entre si, muitas das vezes através de protocolos de comunicação muito diferentes. Isto tem levado a que atualmente exista uma cooperação muito limitada entre os três níveis que na prática se traduzem por *Enterprise Resource Planning* (ERP) para o nível do planeamento, *Manufacturing Execution System* (MES) para o nível da monitorização e nível dos dispositivos distribuídos que correspondem a execução das tarefas.

A utilização da arquitetura orientada a serviços (SOA) através do uso dos serviços *Web* permite a adoção de uma tecnologia unificadora para todos os níveis de uma organização. Esta arquitetura é formada por componentes modulares onde está descrita a lógica das operações referentes a uma etapa do processo produtivo. Através da adoção deste paradigma nas suas infraestruturas, as empresas encontraram uma forma rápida de adaptarem os seus processos produtivos ao ambiente de competitividade em mudança constante. A utilização das tecnologias abertas e standard dos serviços *Web* fornecem as ferramentas essenciais para a implementação destas arquiteturas.

A convergência na direção do paradigma SOA contribui para o melhoramento da reatividade e desempenho dos processos industriais, tais como produção, logísticos, ente outros. Isto vai levar a que a informação esteja disponível em tempo real em todos os níveis de uma empresa permitindo a integração entre todos os níveis.

Numa organização, o planeamento e sequenciamento são duas das tarefas mais importantes num sistema de produção. Estas tarefas influenciam fortemente a rentabilidade da produção de um produto e a utilização dos recursos existentes.

No presente trabalho são definidos alguns recursos que pretendem aliar as vantagens da adoção de uma arquitetura SOA, através de serviços *Web* para promover a integração e flexibilidade, ao mesmo tempo que é desenvolvido um método de otimização para o sequenciamento das ordens de produção para o caso de estudo de uma empresa. A otimização é levada a cabo através de algoritmos genéticos cujos resultados obtidos são encorajadores quer em termos de qualidade da solução quer em tempo de execução.

Keywords Web Services; Service Oriented Architecture (SOA); Flexibility; Integration; Production Scheduling; Genetic Algorithm

Abstract

Modern organizations are characterized by the demand for greater flexibility and interoperability. Within the organization can be found a large number of systems from computer applications covering the entire top of the distributed devices to the lower levels of the shop floor level. At the same time organizations also seek greater integration of their information systems to the devices and equipment of the lower levels. The integration of these systems has brought new obstacles due to differences in the way all the elements communicate with each other, often through very different communication protocols. This has meant that there is now a very limited cooperation between the three levels which in practice translates for Enterprise Resource Planning (ERP) to the level of planning, Manufacturing Execution System (MES) for the level of monitoring and level of distributed devices that match the tasks..

The use of service-oriented architecture (SOA) through the use of web services allows the adoption of a unifying technology for all levels of an organization. This architecture consists of modular components which are described by logical operations pertaining to a stage of the production process. By adopting this paradigm for their infrastructures, companies have found a quick way to adapt their production processes to the competitive environment in constant change. The use of open standard technologies and web services provide the essential tools to implement these architectures.

The convergence towards the SOA paradigm contributes to improving responsiveness and performance of industrial processes such as production, logistics, other entity. This will mean that the information is available in real time at all levels of a company allowing the integration of all levels.

In an organization, the planning and scheduling are two of the most important tasks in the production system. These tasks strongly influence the profitability of production of a product and the use of existing resources.

In the present study some features that aim to combine the advantages of adopting a SOA through Web services to promote integration and flexibility at the same time which is developed an optimization method for the sequencing of the production orders in the case are set study of a company. The optimization is carried out using genetic algorithms whose results are encouraging in terms of quality of solution or at runtime.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Siglas e Acrónimos

AG	Algoritmo Genético
ASP	*Active Server Page*
API	*Application Programming Interface*
ARIS	*Architecture of Integrated Informations System*
BEEP	*Blocks Extensible Exchange Protocol*
CAD	*Computer Aided Design*
CAM	*Computer Aided Manufacturing*
CAPP	*Computer Aided Process Planning*
CNC	*Computer Numerical Control*
DARPA	*Defence Advanced Research Projects Agency*
DCE	*Distributed Computing Environment*
CIM	*Computer Integrated Manufacturing*
CIMOSA	*Computer Integrated Manufacturing Open System Architecture*
CR	*Critical Ratio*
DCOM	*Distributed Component Object Model*
DPWS	*Device Profile for Web Services*
CORBA	*Common Object Request Broker Architecture*
EDD	*Earliest Due Date*
ERP	*Enterprise Resource Planning*
FCFS	*First Come, First Served*
FMS	*Flexible Manufacturing System*
FTP	*File Transfer Protocol*
HTML	*Hipertext Markup Language*
HTTP	*HiperText Transfer Protocol*
IP	*Internet Protocol*
JIT	*Just In Time*
LPT	*Longest Processing Time*
MES	*Manufacturing Execution System*

NP	Não Polinomial
OPC	*OLE for Process Control*
PLC	*Programmable Logic Controller*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Calls*
SCADA	*Supervision Control And Data Aquisition*
SI	Sistemas de Informação
SIRENA	*Services Infrastructure for Realtime Embedded Network Applications*
SMTP	*Simple Mail Transfer Protocol*
SOA	*Service Oriented Arquitecture*
SOA4D	*Service Orientes Architecture for Devices*
SOAP	*Simple Object Acess Protocol*
SOCRADES	*Service-Oriented Cross-Layer Infrastructure for Distributed Smart Embedded Systems*
SPT	*Shortest Processing Time*
TCP	*Transmission Control Protocol*
TI	Tecnologias de Informação
UA	*Unified Architecture*
UBR	*UDDI Business Regestry*
UDDI	*Universal Description, Discovery and Integration*
UPnP	*Universal Plug and Play*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
XML	*eXtended Markup Language*
WS4D	*Web Services for Devices*
WSDL	*Web Service Description Language*
W3C	*World Wide Web Consortium*

Capítulo 1

Introdução

1.1 Contextualização

Muito se tem falado da crise da zona Euro, da globalização económica e de como as empresas devem participar ativamente neste mercado inovador. Porém, o processo de mudanças pelo qual não só as grandes empresas passam mas também as pequenas e médias empresas estão a passar para conseguirem manter a competitividade internacional é bastante difícil e percebe-se um ritmo frenético da procura por flexibilidade tecnológica e de gestão.

No passado, a preocupação das empresas era produzir em massa *(mass production)*, um conceito que foi introduzido no início do século 20 por Henry Ford, caracterizado por uma produção do mesmo produto em grande escala. Durante vários anos, este paradigma foi largamente aceite e implementado. Porém, nas últimas décadas, alterações nos mercados e o aumento da concorrência fez com que as empresas precisassem de alterar os seus métodos produtivos. A produção em massa não é capaz de responder aos desafios de um mercado moderno, dinâmico e global. A globalização dos mercados trouxe às organizações da área da produção novos requisitos para se manterem competitivas, tais como redução de preços, produtos de melhor qualidade, redução dos tempos de fabrico e entrega, diversidade na oferta dos produtos, ente outras. A rigidez da produção em massa, incapaz de lidar com variações de produtos no processo de fabrico, tornou-se obsoleta e com o mercado global, este paradigma de produção manteve-se viável para apenas alguns tipos de produtos.

Para responder às necessidades de reduzir lotes de fabrico surgiu o paradigma da produção personalizável em massa *(mass costumization)*. Este paradigma tem como principais características a elevada diversidade e complexidade de produtos, procura variável e por vezes imprevisível e um ciclo de vida dos produtos mais reduzido. Na atualidade, cada produto pode ter diferentes modelos onde cada modelo pode ser personalizável de forma a responder às necessidades dos clientes, exemplo que se pode encontrar na industria automóvel.

Contudo a alteração do paradigma de produção não se torna suficiente e na procura da redução do risco de se tornarem menos competitivas, as organizações viraram-se para a utilização de novas tecnologias e novos conceitos de produção nas suas atividades produtivas. Nos anos 80, uma companhia japonesa introduziu no seu processo de planeamento um novo paradigma denominado *Lean Manufacturing*, cuja a ideia principal passa pela redução do tempo entre o pedido do cliente à entrega do produto através da

redução de desperdícios. A filosofia *Lean* é uma extensão do conceito *Just In Time* (JIT) que consiste em ter os materiais corretos no local correto e no tempo certo, eliminando ao máximo a necessidade da existência de stocks.

Este ritmo de mudanças em todas as áreas está a impulsionar a mudança dos processos de informação para níveis mais elevados. Atualmente torna-se cada vez mais premente conseguir organizar e gerir os sistemas de produção, definindo estratégias adequadas para ambientes e requisitos de mercado diferenciados. Desta forma, é fundamental conseguir identificar as principais funções de planeamento e controlo da produção e a sua aplicabilidade nos sistemas de produção.

No campo das novas tecnologias o recurso às Tecnologias de Informação (TI) e sistemas de automação é cada vez maior.

Torna-se claro que as empresas dirigem-se para uma infraestrutura dominada pela heterogeneidade [1] de onde advêm os problemas de integração, essencialmente devido aos protocolos de comunicação distintos de cada recurso levando muitas vezes à obrigatoriedade de utilizar *softwares* ou implementações proprietárias de cada fornecedor de equipamento.

No entanto, e de forma a obter informações atualizadas e ser capaz de reagir a mudanças das condições de uma forma flexível e ótima, é necessária a circulação de informação em tempo real através de todos aos níveis da empresa, ou seja, desde o chão da fábrica até ao níveis superiores de decisão. De forma a caminhar nesse sentido, cada vez mais as empresas começam a adotar infraestruturas direcionadas a uma arquitetura orientada a serviços [1].

A motivação para este trabalho passa pela identificação e implementação de métodos e tecnologias que permitem a integração entre os diferentes níveis de uma empresa, tendo em conta os últimos paradigmas que estão a ser adotados pela indústria, ao mesmo tempo que é adotado um método para a otimização do sequenciamento da produção para o caso prático de uma empresa.

Para uma visão um pouco mais concreta sobre o cenário vigente nesta empresa, a sua atividade inicia-se através de um conjunto de encomendas que, uma vez aceites são discutidas e planeadas a cada sexta-feira para iniciarem a produção na segunda-feira da semana seguinte. Uma das maiores preocupações da empresa é de como realizar um plano de produção para não existirem atrasos nas entregas dos pedidos e tirar o máximo partido da sua capacidade produtiva numa industria de pequeno/médio porte no segmento da produção de fechaduras e acessórios.

O planeamento da produção estende-se à satisfação do cliente, o que obriga a empresa a dominar os seus fluxos de informação e processos de produção. Desta forma, criar uma ferramenta de informação de apoio ao planeamento da produção, que seja coerente com os objetivos estratégicos e táticos da empresa, permitirá uma maior afirmação e melhor funcionamento e capacidade de resposta da mesma aos desafios que diariamente enfrenta, por forma a estar o melhor possível, preparada para conhecer e responder às necessidades e expetativas dos clientes.

Este primeiro capítulo tem por finalidade transmitir ao leitor uma visão geral da dissertação, tendo-se começado pela contextualização e os objetivos gerais definidos para este trabalho, bem como os procedimentos utilizados. Em seguida são apresentadas

[1] Entende-se por heterogeneidade o uso de equipamentos, dispositivos e/ou sistemas operativos de diferentes fabricantes que possuem os seus próprios protocolos de comunicação entre equipamentos do mesmo fabricante.

informações mais detalhadas, nomeadamente acerca das tecnologias que pretendem trazer a solução aos problemas da integração assim como um potencial método de otimização do planeamento da produção da empresa presente no caso de estudo.

1.2 Relevância do tema

A crescente competitividade e globalização do mercado tem obrigado as empresas a procurarem novas soluções para os desafios constantes que aparecem e conceitos para sistemas de produção, com vista ao aumento da flexibilidade, adaptabilidade, capacidade de cooperação e de integração [2]. A necessidade do aumento da produtividade e da qualidade na produção de produtos revolucionou o desenvolvimento industrial, visando a utilização de novas Tecnologias de Informação (TI) na produção, no sentido de aumentar a eficiência e a flexibilidade dos equipamentos e dos processos industriais [3].

A procura do controlo e flexibilização da informação levou a que a integração entre os diferentes sistemas se torna-se uma das grandes prioridades organizacionais. A constante evolução das TI trouxe a criação de realidades tecnológicas diferenciadas e o aumento da heterogeneidade de soluções no mercado tornando a questão tecnológica da integração de Sistemas de Informação (SI) cada vez mais complexa, sendo hoje um autêntico desafio quanto à sua flexibilidade, adaptabilidade, implementação, manutenção e gestão [4].

No âmbito dos sistemas de produção, o aparecimento e a evolução de novas ferramentas baseadas na *Web* abrem novos horizontes para a resolução de problemas clássicos de integração de sistemas, cooperação e descentralização [2].

Um sistema industrial está constantemente sujeito a mudanças quer de produtos a produzir quer de novos recursos que necessitam de ser integrados no sistema o que leva à existência de uma grande heterogeneidade entre sistemas utilizados. Daí advêm os problemas de integração, essencialmente devido aos protocolos de comunicação distintos de cada recurso, levando muitas vezes à obrigatoriedade de utilização de aplicações informáticas ou implementações proprietárias de cada fabricante de equipamento [5].

No sentido de facilitar a integração e melhorar a flexibilidade e adaptabilidade dos sistemas de produção tem-se verificado nos últimos anos a adoção de uma arquitetura SOA. O uso dos serviços *Web* no sentido de implementar uma arquitetura SOA trouxe vantagens ao nível da comunicação entre processos devido ao facto destes serviços assentarem na linguagem *eXtensive Markup Language* (XML) o que os torna independentes do tipo de linguagem de cada recurso. Desta forma, os serviços *Web* permitem de uma forma simples efetuar a integração dos diferentes níveis de uma organização permitindo características tais como a interoperabilidade [2], usabilidade e capacidade de reutilização [5].

Um requisito essencial das organizações modernas passa por estarem equipada por um sistema de informação robusto, que é capaz de comunicar com todos os dispositivos e componentes programáveis. Em geral, o problema da integração de diferentes equipamentos na industria torna-se um tarefa complicada devido à existência de uma vasta variedade de equipamentos de diferentes fabricantes, cada um com o seu protocolo de comunicação proprietário.

[2]Interoperabilidade é a capacidade de um sistema de comunicar de forma transparente com outro sistema (semelhante ou não).

1.3 Objetivos

No contexto desta dissertação, pretende-se propor novos serviços *Web* que permitam a rápida e fácil integração de novos recursos e/ou métodos na instalação fabril. Serviços *Web* esses que simplificam a integração de capacidades de planeamento e que permitem uma melhor interação entre os diversos sistemas presentes em cada nível da produção.

Subsequentemente o objetivo consiste em desenvolver um método de planeamento do sequenciamento das ordens de produção de uma empresa segundo duas vertentes, a primeira de acordo com regras de prioridades estabelecidas pela empresa e a segunda segundo um método de otimização.

Por fim, proceder a uma análise comparativa entre os resultados dos respetivos métodos de sequenciamento desenvolvidos.

1.4 Solução proposta

No âmbito deste trabalho foram analisadas as tecnologias mais promissoras para a integração dos sistemas de produção.

Para promover a integração, eficiência e colaboração entre sistemas de processos de produção, o presente trabalho apresenta o desenvolvimento de serviços *Web* que vão de encontro ao paradigma mais atual da arquitetura SOA. Os serviços desenvolvidos procuram apresentar uma solução ao caso de estudo onde se procura fazer o planeamento e sequenciamento de ordens de produção de uma empresa. São apresentadas duas soluções, uma que faz o sequenciamento de acordo com regras de prioridades estabelecias pela empresa e outra que procede à otimização do sequenciamento através do uso de um algoritmo genético. O algoritmo de otimização é aplicado para encontrar o melhor sequenciamento de ordens que minimiza o problema dos trabalhos atrasados com pesos atribuídos, problema que simula o ambiente de produção da empresa e ao mesmo tempo fornece uma aplicação prática dos métodos de otimização num caso real.

Posteriormente, é desenvolvida uma aplicação *Web* que além de servir de interface gráfica para a apresentação dos resultados, pretende demonstrar as facilidades de invocação dos serviços *Web*.

1.5 Organização da Dissertação

O capítulo 2 diz respeito à revisão bibliográfica sobre os temas principais na área de integração de sistemas de produção, nomeadamente os seus problemas e arquiteturas que podem ser utilizadas na procura de soluções aos problemas encontrados. São descritas algumas tecnologias e especificações existentes e a forma como estas podem ser aplicadas no âmbito deste trabalho.

O capítulo 3 aborda temas como como as tecnologias utilizadas, particularmente infraestruturas de suporte tecnológico e protocolos de comunicação para a transferência de informações.

No capítulo 4 descrevem-se outros fundamentos teóricos de suporte à realização desta dissertação, de entre os quais merecem particular atenção os algoritmos genéticos, o método a ser utilizado na obtenção da otimização do problema de sequenciamento.

O capítulo 5 é dedicado ao sistema desenvolvido, através de uma visão geral da sua estrutura e funcionamento bem como a sua implementação onde são apresentadas e clarificadas as etapas concretizadas para o desenvolvimento do presente trabalho.

No capítulo 6 apresentam-se as principais conclusões, e discutem-se os resultados obtidos na aplicação dos serviços *Web* desenvolvidos assim como os resultados comparativos entre os dois métodos se sequenciamento das ordens de produção. Incluem-se ainda algumas recomendações e propostas de desenvolvimento de trabalho futuro.

Por fim, apresentam-se as referências bibliográficas que fundamentaram e apoiaram a realização deste trabalho.

Capítulo 2

Revisão do Estado da Arte

Na atual Era da Informação, os consumidores tornaram-se cada vez mais sofisticados, exigindo produtos com níveis elevados de qualidade, a preços reduzidos com tempos de produção menores, o que requer menor tempo de entrega. Adicionalmente, no mercado atual, fatores pouco contabilizados tais como a qualidade, design do produto, inovação e serviços de entrega, tornam-se cada vez mais determinantes para o sucesso do produto neste mercado cada vez mais global.

No ano 1998, um grupo dos Estados Unidos da América apresentou uma visão do ambiente de competitividade para as organizações do futuro onde identificaram as forças técnicas, políticas, socais e económicas mais importantes na obra *"Visionary Manufacturing Challenges For 2020"* [6]. Os elementos deste grupo identificaram os seguintes aspetos mais importantes:

- Clientes sofisticados exigirão produtos com maior grau de personalização de forma a irem de encontro às suas necessidades,
- Respostas rápidas a alterações no mercado tornar-se-ão necessárias para a sobrevivência das organizações num clima de competitividade, melhorado pela comunicação e partilha de informação,
- Criatividade e inovação serão requisitos em todos os aspetos das organizações para se manterem competitivas,
- Desenvolvimentos em tecnologias inovadoras de processos de fabrico, modificações da extensão e escala dos processos produtivos,
- Questões ambientais serão mais predominantes, uma vez que o ecossistema será marcado pelo crescimento populacional e pela emergência de novas economias de alta tecnologia.

2.1 O problema da integração

As organizações modernas são caraterizadas pela necessidade crescente de flexibilidade e interoperabilidade. Dentro de uma organização podem ser encontrados os mais variados tipos de equipamentos e aplicações que cobrem toda a hierarquia do ambiente industrial [7]. A crescente necessidade de integração dos sistemas de informação deriva

de fatores relacionados com a evolução das organizações, dos mercados e da tecnologia. A necessidade de mostrar na Internet alguma informação que se encontra nos sistemas informáticos internos, a necessidade de partilhar informação entre sistemas heterogéneos ou a automatização de processos organizacionais são algumas das situações que necessitam de soluções especificas. Estas necessidades surgem normalmente no dia-a-dia das organizações devido à sua dinâmica e à constante evolução tecnológica. Esta é uma realidade complexa, onde existe uma forte competitividade ao nível das soluções e onde as organizações facilmente não encontram as respostas mais adequadas às suas reais necessidades [4].

Do ponto de vista organizacional, a integração preocupa-se em facilitar a troca de informação, controlo do fluxo de materiais dentro dos limites da organização através da ligação entre todas as funcionalidades necessárias e as funcionalidades de entidades heterogéneas (sistemas de informação, dispositivos, aplicações e pessoas) com o intuito de promover a comunicação e cooperação de forma que a organização funcione como um todo. Conseguindo ser alcançada, pode ser esperado um aumento na produtividade geral, na flexibilidade e capacidade para lidar com alterações ou modificações (reatividade) [8].

O modelo mais presente na literatura estabelece a hierarquia de uma empresa como sendo constituída por quatro níveis diferentes: o nível do campo, o nível do chão da fábrica, nível de decisões operacionais e o nível de planeamento, logística e decisões estratégicas.

A integração entre os sistemas de informação e os sistemas de produção é um dos maiores objetivos a ser realizado nos sistemas constituintes de uma empresa [9]. Os *Manufacturing Execution Systems* (MES) fornecem informação em tempo real sobre os acontecimentos no chão da fábrica quer para gestores (sob um ponto de vista estratégico), como para operadores (sob uma perspetiva puramente operacional). Trata-se também de um sistema que faz a ligação entre os sistemas de planeamento superiores usados no planeamento estratégico, tais como *Enterprise Resource Planning* (ERP) e os sistemas de aquisição e controlo de dados SCADA. A grande importância dada ao sistema MES reside em grande parte às funcionalidades que permite e à sua iteração com os dispositivos encontrados no ambiente fabril. As funcionalidades principais incluem o planeamento da produção, gestão das ordens de produção, gestão das máquinas e planos de manutenção, rastreabilidade do processo produtivo, monitorização do trabalho em progresso, monitorização dos inventários e documentação [10].

A integração pode ser aplicada verticalmente e horizontalmente dentro da organização. A integração vertical refere-se à linha de integração entre o nível superior de administração até ao nível inferior do chão da fábrica. A integração horizontal refere-se à integração dos vários domínios (áreas de cooperação ou processos) dentro da empresa ou até com parceiros e os seus ambientes produtivos.

O tema da integração entre os níveis de uma empresa tem vindo a ser estudado deste 1985 sob o conceito *Computer Integrated Manufacturing* (CIM). Este conceito surgiu pela primeira vez em 1973 na obra com o mesmo titulo pela mão do Dr. Joseph Harringon. Na sua obra, o autor define este conceito como uma estratégia de integração entre todas as tecnologias de produção e os sistemas de organização através do uso de computadores e sistemas informáticos [11]. Ente os trabalhos mais conhecidos sobre a integração destacam-se *Computer Integrated Manufacturing Open System Architecture* (CIMOSA) [12] onde surge pela primeira vez o conceito de arquitetura da empresa e *Architecture of Integrated Informations Systems* (ARIS) [13]. As arquiteturas propostas

pelos autores são direcionadas para o desenvolvimento e implementação dos sistemas de uma empresa [8].

Os sistemas de comunicação têm vindo a ser introduzidos em vários níveis da automação tendo em mente permitir uma conexão entre dispositivos de uma forma flexível e compreensível. Apesar da ideia inicial do conceito CIM passar pela transmissão de informação de forma transparente em todo o sistema de automação, a evolução das redes de comunicações industriais tomou outros caminhos na prática. Na prática os domínios das organizações não são abordados de uma forma integrada. Cada domínio possui a sua própria linguagem, os seus modelos e utiliza as suas próprias técnicas e ferramentas. A comunicação e tomada de decisões através de todos os domínios mostra-se seriamente debilitada. Apesar de existirem no mercado algumas ferramentas que fornecem funcionalidades de comunicação entre os níveis da organização, em geral possuem um fraco suporte e não se integram com outras ferramentas além de não permitirem configurações suficientes no contexto de cada empresa. Porém, não é realista supor que as companhias de desfaçam das sua práticas e ferramentas existentes por abordagens completamente novas. Em vez disso a integração deve focar-se em permitir a interação entre as técnicas e ferramentas existentes através da integração segundo um nível apropriado de abstração [14].

A implementação de uma *Service Oriented Architecture* é reconhecida como uma abordagem promissora como suporte à integração e colaboração dos recursos distribuídos. Esta arquitetura permite definir os recursos distribuídos através de vários elementos denominados serviços que podem ser invocados de forma independente quer por consumidores de serviços externos como internos para executar o processamento de funções simples ou então trabalhar em conjunto através da conjugação de serviços para formar novas funcionalidades em processos já existentes [15].

2.2 Arquitetura Orientada a Serviços

A arquitetura SOA pode ser descrita como um paradigma que traduz os módulos de um sistema de informação em serviços que podem ser organizados em conjuntos para formar processos.

Sem dúvida alguma, as organizações modernas são cada vez mais dependentes das suas TI. Consequentemente é exigido das TI o mesmo que a uma organização que pretende manter-se competitiva no atual mercado de constantes mudanças. Para isso é exigido que possuam um elevado grau de flexibilidade e agilidade para responder às mudanças.

Com o crescimento das organizações existe uma constante necessidade de integração de novos equipamentos ou reorganização dos já existentes onde a arquitetura SOA torna-se uma solução ideal para trabalhar com sistemas complexos e distribuídos. Apesar de existir um suporte alargado de grandes empresas tecnológicas como por exemplo a IBM, HP, Microsoft e SUN, a arquitetura SOA e os seus benefícios ainda não se encontram bem definidos [16]. Na bibliografia podem ser encontradas várias definições para este paradigma. De seguida são apresentadas três:

- "SOA consiste num conceito empresarial, uma ideia ou uma abordagem de como as funcionalidade das TI podem ser planeadas, desenhadas e implementadas em formato modular e ir de encontro às necessidades específicas"[17].

- "A arquitetura SOA consiste numa arquitetura de *software* baseada em aplicações em servidores, serviços, repositórios de serviços e camadas de transmissão desses serviços onde um serviço consiste num contrato com uma ou mais interfaces e/ou implementações"[18].

- "Uma arquitetura SOA é um paradigma para a organização e utilização de capacidades distribuídas que podem estar sob o controlo de outros domínios de utilização"[19].

Apesar das diferentes interpretações desta arquitetura em todas está presenta a utilização de uma orientação a serviços para a sua implementação.

Para o presente trabalho, faz-se uso da terceira definição que trata a arquitetura SOA como uma arquitetura que faz uso de capacidades distribuídas, permitindo o controlo dessas capacidades ao longo de todos os níveis de uma empresa.

A chave é a flexibilidade. Para as grandes empresas e para os grandes sistemas distribuídos, a flexibilidade nas TI tornou-se uma das áreas de atuação para a criação de valor acrescentado.

Contudo, os sistemas e os processos tornam-se cada vez mais complexos resultante da evolução da fase em que a automação se baseava em ilhas para um enorme sistema distribuído. Atualmente o desafio encontra-se na manutenção desses sistemas.

O crescente recurso a novos equipamentos dos mais variados construtores obriga a uma nova abordagem que aceite a heterogeneidade e conduza à descentralização.

A arquitetura SOA é uma abordagem que ajuda os sistemas a manterem a escalabilidade[3] e flexibilidade, ao mesmo tempo que estes crescem em complexidade e também ajuda a criar uma ponte entre as TI. Arquitetura SOA é um termo que representa um modelo no qual a lógica de automação é decomposta em unidades de lógica mais pequenas e distintas. Coletivamente, estas unidades representam uma grande parte da lógica de automação de um processo. Individualmente, estas unidades de lógica podem ser distribuídas.

SOA encoraja as unidades de lógica individuais a existirem de forma autónoma e independente, contudo não isoladas umas das outras. Estas unidades lógicas são também conhecidas como serviços. Do ponto de vista de [20], a coletiva de lógica que definem os seus processos para avançar e evoluir no tempo está em mudança em resposta às influências internas e influência externas. Das TI, toda a lógica de uma empresa pode subdividir-se em dois grupos principais: lógica operacional e lógica de aplicações (figura 2.1).

[3]Considera-se escalabilidade como um sistema ou uma rede com capacidade de ser aumentada ou atualizada em pouco tempo, que pode ser através da adição de *software* ou *hardware* extra com o objetivo de receber o trabalho extra sem afetar a *performance* do sistema global.

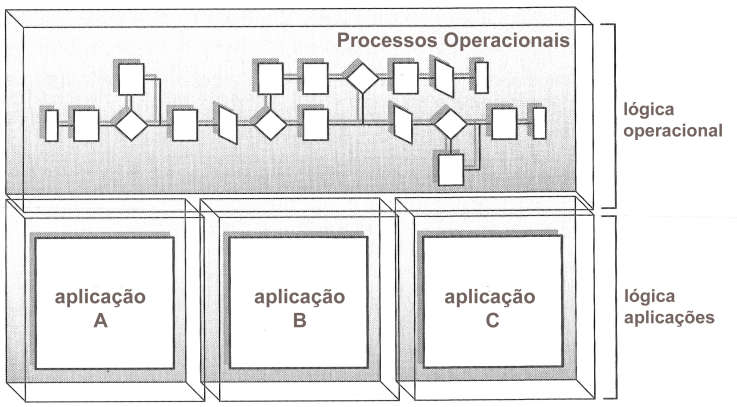


Figura 2.1: Os dois maiores grupos de lógica numa empresa (adaptada de [20])

A lógica operacional consiste numa implementação documentada dos requerimentos originados das áreas de negócio da empresa. Esta lógica encontra-se organizada num conjunto de processos de forma a expressar os requerimentos pretendidos. Por sua vez, a lógica de aplicações mostra-se como a implementação da lógica operacional através do recurso a soluções tecnológicas de automação e informação. A lógica pode expressar fluxos de trabalho e processos através da compra ou desenvolvimentos de novos componentes dentro das fronteiras da infraestrutura das tecnologias de informação, dependências de segurança e capacidades técnicas. A orientação a serviços vem introduzir um novo conceito que permite melhorar a forma como a lógica é representada, vista, trabalhada e partilhada através do conceito de serviços. Os serviços podem ser formados por diferentes tipos de lógica e cada lógica para um determinado contexto que pode ser específico de uma tarefa de negócio, uma entidade ou outro tipo de ação lógica (figura 2.2).

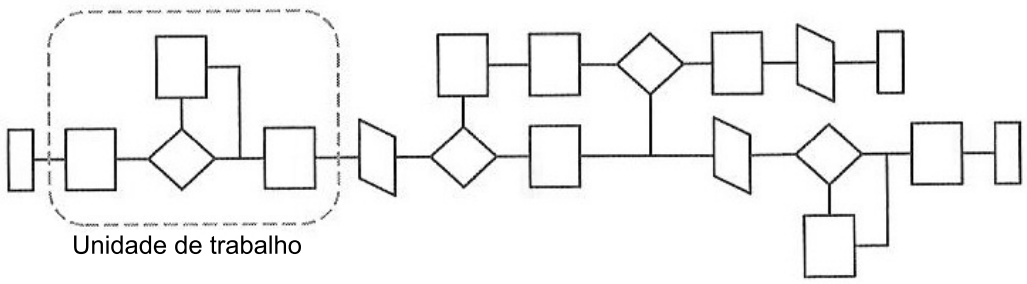


Figura 2.2: Pormenor de uma operação de um processo (adaptada de [20])

Um serviço representa um grupo de operações lógicas capazes de efetuarem as unidades de trabalho e um processo inclui as regras de negócio que determinam quais os serviços a serem utilizados em cada operação de forma a completar a unidade de automação. Por outras palavras, um processo representa uma grande parte de trabalho que requer conclusão através de unidades de trabalho mais pequenas (figura 2.3).

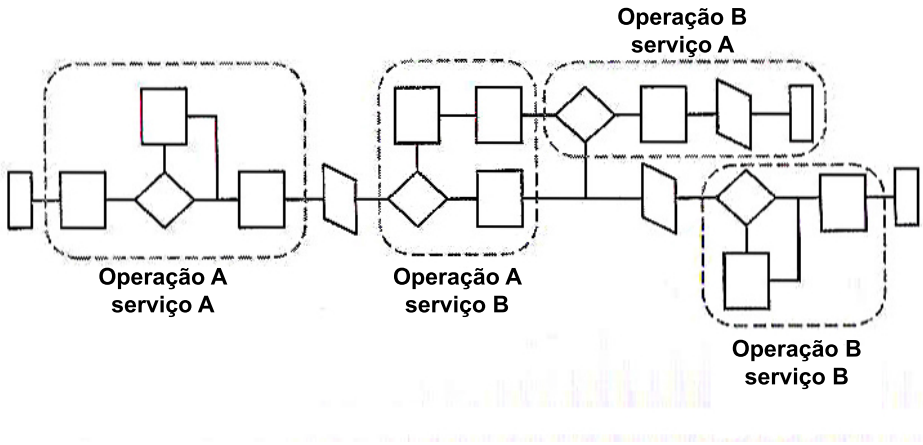


Figura 2.3: Operações pertencentes a diferentes serviços representando várias partes do processo lógico (adaptada de [20])

Os serviços também podem ser constituídos por um ou mais serviços. Cada serviço pode conter uma tarefa de um passo individual a um sub-processo composto por diferentes passos intermédios. Um serviço pode ainda incorporar o processo lógico inteiro [20].

Contudo a utilização da arquitetura SOA não tem de ser necessariamente exclusiva dos níveis superiores da organização. Nos últimos anos têm sido feitos estudos sobre a sua aplicação ao nível dos processos de produção.

Uma das primeiras aplicações da arquitetura SOA para a integração dos níveis inferiores apareceu em 2003 através do projeto *Services Infrastructure for Realtime Embedded Network Applications* (SIRENA)[21], um projeto europeu que decorreu ente 2003 e 2005, gerido por um consórcio onde estava incluída a *Siemens* e a *Scheiner Electric*, entre outras entidades. O projeto teve como objetivo a criação de uma arquitetura orientada a serviços que permitisse a conexão simples entre dispositivos heterogéneos desde a industria automóvel, comunicações ou domótica, dispositivos com limitações de processamento e memória mas com suporte de comunicações de rede no âmbito da automação industrial. Deste projeto resultou a infraestrutura "SIRENA *Framework*"criada para facilitar o desenvolvimento, integração, manutenção e controlo dos dispositivos e serviços contidos numa interface SIRENA através da adoção da especificação *Device Profile for Web Services* (DPWS) que define os requisitos mínimos para a implementação de um serviço *Web* [22]. Do projeto SIRENA resultaram ainda outras especificações como *SOA for Devices* (SOA4D) e *Web Services for Devices* (WS4D).

A arquitetura do projeto SIRENA continuou a ser desenvolvida através do projeto *Service-Oriented Cross-Layer Infrastructure for Distributed Smart Embedded Systems* (SOCRADES). O objetivo principal do projeto SOCRADES passa pelo desenvolvimento de ferramentas e métodos que consigam alcançar flexibilidade, reconfiguração, escalabilidade e interoperabilidade entre sistemas embebidos distribuídos e descentralizados dentro de uma rede de comunicação. Com essa finalidade o projeto faz uso da arquitetura SOA para implementar a integração tanto ao nível inferior do chão da fábrica quer ao nível superior das aplicações industriais [23]. O projeto implementou com sucesso a arquitetura SOA focando-se em três grandes grupos tecnológicos: uma plataforma com os serviços desenvolvidos, a integração empresarial, infraestrutura de sensores/atuadores

de tecnologia *wireless* em rede. No primeiro caso a plataforma de serviços permite garantir que os serviços dos processos podem ser facilmente reconfiguráveis através da troca de dispositivos, sendo que cada um expõem serviços prontos para serem consumidos. Assim, cada vez que um dispositivo ou componente é introduzido no sistema, o novo dispositivo é imediatamente reconhecido pelos restantes e pelo processo. Isto implicada um registo e identificação eficiente do componente na infraestrutura global. O segundo aspeto abordado diz respeito à comunicação do processo de automação. Os dispositivos que são removidos e adicionados podem comunicar entre si através através de redes de comunicação com fios ou sem fios. A terceira área tecnológica envolvida no projeto diz respeito à uniformização de protocolos que envolvem os níveis superiores de uma empresa com os seus níveis mais baixos, significando que as camadas superiores seriam capazes de obter informações diretamente dos processos do chão da fábrica.

Em [24], o autor apresenta uma arquitetura SOA com o objetivo de resolver os problemas da interoperabilidade ente os diversos componentes de um processo de automação, mais especificamente na comunicação entre os recursos de produção e os sistemas de comunicação. É apontada a utilização de tecnologias abertas como a principal vantagem da arquitetura SOA em comparação com outros sistemas como o *OLE for Process Control*(OPC). Com o trabalho, o autor pretende mudar os padrões da industria através da alteração da programação distribuída orientada a objetos para a programação distribuída orientada a serviços.

No caso de [25] são abordadas arquiteturas SOA para a monitorização e controlo de células robotizadas industrias. São apresentadas ferramentas informáticas para a implementação de duas plataformas: *Universal Plug and Play* (UPnP) e *Descentralized Software Services Protocol* (DSSP) baseados em linguagem SOAP para a plataforma *Microsoft Robotics Studio*.

2.3 Serviços *Web*

No ambiente altamente competitivo entre empresas da atualidade, a habilidade de permitir a troca de informação de forma eficiente tornou-se um requisito fundamental. As organizações tem de se tornar capazes de integrar a cadeia de informação interna e dos processos através da integração vertical e horizontal. Os Serviços *Web* tratam-se de uma tecnologia emergente da arquitetura orientada a serviços para sistemas distribuídos[26].

Segundo o consórcio W3C, responsável pela criação do standard destes serviços, "um serviço *Web* é um sistema de *software* projetado para suportar uma iteração entre máquinas interoperáveis dentro de uma rede de comunicação. Possui uma interface descrita num formato processável por uma máquina, especificamente *Web Service Descriptive Language* (WSDL). Os outros sistemas interagem com o serviço *Web* de uma forma definida pela sua descrição através de mensagens SOAP, tipicamente através do protocolo *HyperText Markup Language* (HTML) e XML em conjunto com outros padrões *Web*"[27].

Esta abordagem aos sistemas distribuídos não é completamente nova. Os serviços *Web* podem ser comparados a arquiteturas anteriores de sistemas distribuídos, tais como *Object Management Group's Commom Object Request Broker Arquitecture* (CORBA) ou a *Open Group's Distributed Computing Environment* (DCE), Microsoft Enterprise Javabeans ou Microsoft Distributed Component Object Model (DCOM). Em todos estes ambientes de comunicação, as aplicações do lado do servidor são desenhadas para expor

as suas interfaces aos serviços ou funções que oferecem. Os clientes localizam estes serviços através de uma procura de diretório e ligam-se ao serviço desejado utilizando a informação armazenada no diretório para o efeito.

Os serviços *Web* apareceram com o objetivo de encontrar uma forma de fornecer serviços, a aplicações clientes, através da rede de Internet. Contudo, neste tipo de aplicações, não está contemplado o desenvolvimento de interfaces gráficas para o utilizador, tendo em atenção que um serviço *Web* trata-se de um *"Middleware"* (*Software* intermédio) que é integrado nas interfaces gráficas já desenvolvidas como por exemplo um *Browser*[4][2].

Como está ilustrado na figura 2.4, as aplicações clientes têm habilidade de descobrir ou localizar os serviços *Web* através do acesso ao registo UDDI ou outro diretório de serviços. Os documentos WSDL armazenados no registo descrevem as características dos serviços disponíveis e os métodos que podem ser utilizados para a interação com estes. As aplicações clientes procuram por serviços *Web*, como o descrito no documento WSDL, que preenchem requisitos específicos. Assim que o serviço apropriado é descoberto, a aplicação cliente pode comunicar com o serviço encontrado utilizando métodos e interfaces definidas no documento WSDL. O cliente invoca a função remota no servidor que procede ao envio e receção de informação utilizando mensagens SOAP.

A maior diferença entre os serviços web e os seus predecessores reside na sua confiança nos protocolos de Internet e métodos da *Wordl Wide Web* e nos protocolos abertos em vez de protocolos especializados e proprietários que são utilizados nos sistemas anteriores [26].

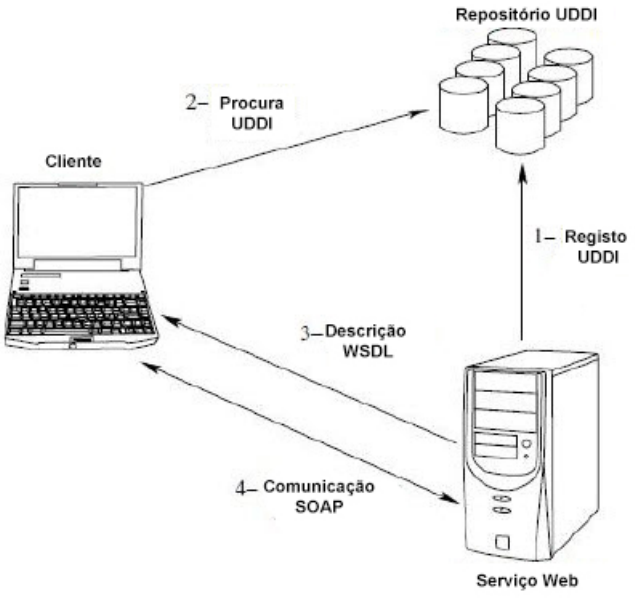


Figura 2.4: Tecnologias utilizadas num serviço *Web* (adaptado de [24])

Assim como os seus predecessores, os serviços *Web* tem o objetivo de permitir a partilha de informações entre aplicações informáticas e partilha de funcionalidade através de redes de comunicação. Os serviços *Web* foram desenvolvidos para tomar partido

[4]Um *Browser* consiste numa aplicação de *software* que recebe e apresenta a informação recebida através da Internet, como por exemplo o Internet Explorer, Google Chrome, entre outros.

da influência dos standards implementados em aplicações de Internet para promover a interoperabilidade através de uma variedade de plataformas.

Os mais recentes desenvolvimentos apresentam esta tecnologia como a solução para os atuais problemas de integrações entre as várias aplicações clientes encontradas num ambiente industrial, bem como aplicações remotas e de recursos pois estes mostram-se independentes da plataforma de desenvolvimento.

Os serviços *Web* são a tecnologia ideal para a comunicação automática entre sistemas. Na sua arquitetura, são formados por uma aplicação local, responsável por disponibilizar e fornecer o serviço, e por uma aplicação cliente que acede ao serviço.

Uma vez que este tipo de programação faz uso de protocolos normalizados, possibilita uma independência da plataforma e linguagem de programação na comunicação entre os diversos equipamentos ou plataformas.

De seguida são enumeradas algumas das vantagens dos serviços *Web* face aos sistemas distribuídos mais antigos orientados a objetos:

- Permite e interação entre aplicações sem a necessidade de intervenção humana;
- Baseia-se em protocolos standard abertos, tais como HTTP, XML, SOAP, WSDL e UDDI;
- São acessíveis como componentes a partir de qualquer lugar na Internet;
- Conseguem funcionar mesmo com *firewalls* e servidores *proxy*, pois as mensagens são passados sob a forma de documentos XML e todo o tráfego HTTP é transmitido pela porta TCP 80;
- Conseguem tirar partido da autenticação e encriptação *Secure Socket Layer*(SSL);
- Combinam a programação orientada a objetos com o programação *Web*;
- São independentes do sistema operativo, da plataforma de desenvolvimento e da linguagem de programação;
- Apresentam os resultados no formato XML;
- Permitem a diminuição da complexidade e custos de integração de sistemas proprietários.

Para a utilização de um serviço *Web* é imperativo definir o modo como se acede ao serviço e que valor o serviço vai devolver. As definições são escritas num ficheiro XML de acordo com a norma WSDL. O arquivo deve ser construido para que os utilizadores do serviço possam entender o funcionamento do serviço *Web*, sendo este de acesso público.

Também podem ser utilizados para a implementação de arquiteturas orientadas a serviços. Neste modelo de arquitetura, os principais requisitos são serviços que são utilizados por outros serviços, modularizando e aumentando a coesão dos componentes da aplicação.

Um dos exemplos clássicos da aplicação dos serviços *Web* pode ser encontrado em [28]. No exemplo tem-se uma empresa de viagens que pretende disponibilizar aos seus clientes a possibilidade de reservarem um pacote de férias completo onde se encontram

incluídas as viagens de comboio/avião, alojamento, aluguer de automóvel, restaurantes, etc, (figura 2.5).

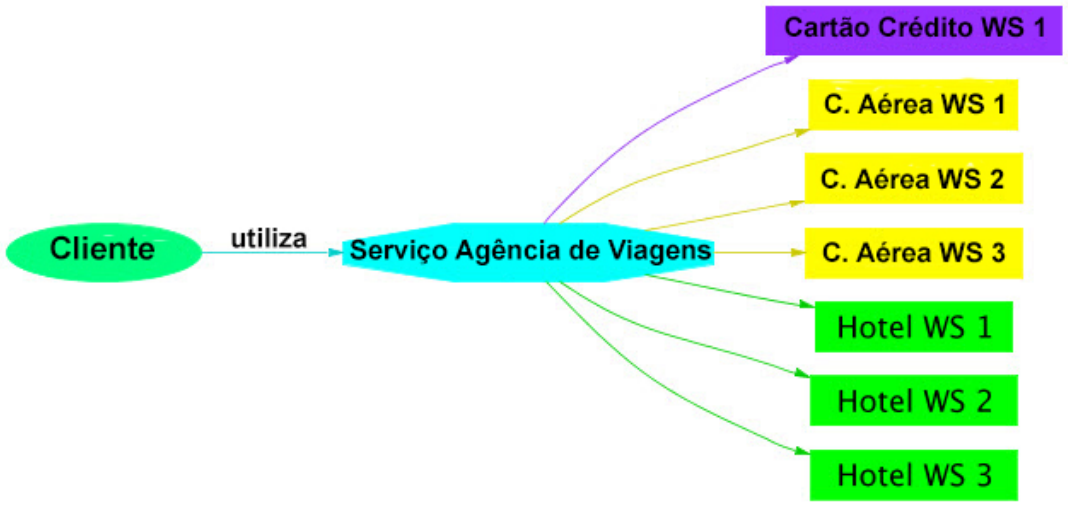


Figura 2.5: Exemplo da agência de viagens (adaptado de [28])

Os serviços externos à empresa de viagens, que executam a reserva do hotel, reserva do voo numa determinada companhia aérea, etc, disponibilizam aplicações informáticas em forma de serviços *Web* que permitem consultar e reservar os seus produtos. As empresas bancárias fazem também o uso dos serviços *Web* que permitem validar os cartões de crédito dos clientes.

Através da especificação UDDI, a empresa de viagens pode encontrar todos os serviços de que necessita para a criação da sua própria aplicação informática. Implementa o acesso aos serviços através da descrição WSDL e para a pesquisa e reserva de produtos faz uso do protocolo SOAP. No entanto o cliente é exigente e gosta sempre de ter algumas opções de escolha. Em resposta à exigência dos clientes a empresa de viagens recolhe as informações introduzidas pelo cliente, destino, data da viagem, número de pessoas, tipos de serviços pretendidos, por exemplo através de um formulário *Web* que automaticamente invoca os respetivos serviços *Web* para disponibilizarem listas de opções que coincidem com os pedidos dos clientes. Após a recolha de todas as informações necessárias, estas são disponibilizadas ao cliente para permitir a escolha da melhor oferta. No caso do cliente selecionar um voo, a aplicação informática invoca novamente o serviço *Web* da companhia aérea para efetuar a reserva. O mesmo processo é repetido para os restantes serviços de forma automática e sem necessidade de intervenção humana.

2.4 Alguns trabalhos desenvolvidos na área

Os serviços Web são uma arquitetura inovadora e possuidora de um enorme potencial para alterar profundamente a forma como as organizações utilizam e trocam informação através dos seus sistemas de informação.

Estes serviços podem ser combinados com aplicações existentes dentro da organização de forma a criar aplicações que podem ser invocadas e localizadas de forma dinâmica

para completar uma determinada tarefa. Estas capacidades dotam as organizações de ferramentas para encapsular aplicações mais antigas dentro de interfaces de serviços *Web* que fornecem o acesso às suas funções. Em [5] o autor propõem um conjunto de serviços *Web* que permitem a aquisição do estado e controlo de um tapete de transporte, acessível através da Internet, que leva a um novo tipo de solução de *Supervision Control and Data Aquisition System* (SCADA), onde se desvia dos protocolos de comunicação mais utilizados hoje em dia e passa a englobar os serviços *Web* como meio de comunicação com um recurso industrial que pode ser acedido de qualquer computador equipados com uma ligação à Internet.

Do ponto de vista de utilização dos serviços *Web* e como o próprio conceito indica, um serviço *Web* desenvolvido por exemplo em linguagem PHP pode ser acedido por uma aplicação cliente em *C#*.

Os utilizadores podem aceder aos serviços *Web* através de *browsers* assim como aplicações cliente em computadores industrias que façam uso dos métodos de invocação destes serviços.

Ao longo dos últimos anos a implementação de automação industrial veio trazer grandes vantagens e inovações à produção industrial, no entanto a complexidade da sua implementação e dificuldade na sua modificação para responder às tendências do mercado tem levado um enorme consumo de tempo e esforço. Em [29] os autores referem a importância da abordagem baseada em componentes. No entanto, e devido à progressão dos sistemas de produção mais eficazes e livres de erros, torna-se necessário complementar a abordagem com novos sistemas de automação e integração da produção que permitam um melhor controlo e monitorização de todo o sistema de produção. Nesse sentido a adoção de uma arquitetura SOA baseada na implementação de serviços *Web* apresenta a capacidade de melhorar todos os aspetos dos sistemas de produção industrial. É implementado com sucesso um serviço *Web* que faz uso da especificação *Device Profile for Web Services*(DPWS) que facilita a comunicação e integração com os equipamentos de baixo nível, nomeadamente sensores e outros dispositivos de controlo.

O autor em [2] faz um estudo de implementação dos serviços *Web* em três possíveis casos de sistemas industriais fazendo uso da arquitetura SOA como infraestrutura de suporte à integração de sistemas de produção distribuídos e descentralizados. Em primeiro, foi abordado um recurso de transporte com controlo por PLC, no segundo caso um centro de maquinagem com controlador *Computer Numerical Control* (CNC) e no último caso o controlo integrado de um *Flexible Manufacturing System*(FMS) formado por vários recursos industriais de transporte e produção, cada um com o seu protocolo de comunicação proprietário.

Em [3] no sentido de permitir a viabilidade de sistemas de monitorização e controlo remoto através dos serviços *Web*, o autor desenvolveu um sistema para aplicações SCADA baseada numa arquitetura descentralizada. O autor desenvolveu diferentes serviços *Web* que comunicam com uma aplicação servidora local, através de um sistema de gestão de bases de dados. A aplicação local faz uso de determinadas rotinas para estabelecer a comunicação entre vários PLC's. Uma vez que os serviços permitem uma abstração dos níveis inferiores a sua integração torna-se mais simples e fácil. Os resultados obtidos mostraram que este tipo de arquitetura pode constituir um potencial acrescido para as mais variadas situações e equipamentos desde ambientes industriais a ambientes académicos e sistemas de automação.

Em [30] é apresentada uma arquitetura para monitorizar e controlar remotamente

um sistema de manufatura. A arquitetura foi desenvolvida com base numa arquitetura orientada a serviços e contém dois componentes principais: o servidor OPC UA [31] e os serviços *Web*. O servidor OPC UA foi desenvolvido através do uso de um adaptador especial como estratégia de migração. Além da especificação básica de acesso à informação, ainda inclui os alarmes a eventos e o historial de dados que são usados para fornecer novas funcionalidades aos serviços *Web*. Uma das principais questões abordadas no trabalho é a gestão de sessões no servidor do serviço *Web* a fim de limitar o número de conexões UA, eliminando os atrasos indesejados. Foram definidos um total de oito serviços *Web* básicos capazes de fornecer informações a todos os níveis da pirâmide de automação. Como resultado, a arquitetura implementada oferece as principais vantagens de uma arquitetura orientada a serviços, a flexibilidade. Os serviços *Web* e a ordem em que são usados nos diferentes cenários podem ser alterados rapidamente levando a menores tempos de reatividade, uma consequência direta da flexibilidade adquirida pelo sistema. Os serviços apresentaram ainda a possibilidade de serem utilizados em novos cenários o que leva à existência de um nível de reutilização maior.

Em [32] é proposta uma metodologia para o controlo e gestão da produção baseada na *Web* de uma empresa virtual. Esta empresa é composta por três sistemas de produção distribuídos e localizados em diferentes cidades. A metodologia proposta inclui o desenvolvimento de um sistema ERP e a sua integração com os módulos de engenharia *Computer Aided Design* (CAD), *Computer Aided Manufacturing* (CAM) e *Computer Aided Process Planning* (CAPP). O sistema ERP é desenvolvido para a *Web* permitindo aos clientes colocarem a sua ordem de produção a partir de qualquer lugar sem a necessidade de ter o equipamento e *software* para realizar o ciclo de desenvolvimento do produto. A metodologia implementada também permite que os operadores de máquinas se possam ligar remotamente ao sistema e executar atividades a partir de qualquer lugar. O cliente faz uso dos serviços de fabricação da arquitetura SOA para executar as operações e os processos necessários para projetar e fabricar a parte desejada de forma eficiente usando ferramentas computacionais para o desenvolvimento do ciclo de vida do produto.

O uso da arquitetura SOA baseada em serviços *Web* ao nível da produção promete a integração vertical e o aumento de interoperabilidade e flexibilidade. Infelizmente existem dois obstáculos principais que dificultam o uso destes serviços neste nível da empresa. Em primeiro lugar tem-se o *software*/*hardware* usado no nível dos sistemas de produção que é diferente do que é usado nos níveis superiores. Em segundo lugar, este nível é mantido pelos operadores e engenheiros de automação que normalmente não estão familiarizados com o uso dos serviços *Web*. Os autores em [33] apresentam o primeiro motor de SOAP para PLC's (SOAP4PLC) para facilitar a introdução dos serviços *Web* ao nível da produção. Este motor oferece um baixo consumo de memória e respeita o menor poder computacional destes dispositivos, permitindo o consumo dos serviços sem a intervenção de um engenheiro de automação.

Em [34] é apresentada uma extensão do motor SOAP4PLC para invocar uma aplicação baseada em serviço *Web* a partir de uma aplicação de PLC. É ainda mostrado um caso de estudo onde a extensão do motor SOAP4PLC é usado para criar a interface da aplicação do PLC responsável pela realização da funcionalidade de uma estação de carregamento de veículos elétricos com um sistema de contabilidade baseado em serviço *Web*.

À medida que avançamos para empresas em tempo real, a comunicação entre todos

os níveis pode levar a um aumento da eficiência. Para alcançar este objetivo, os dispositivos do chão da fábrica têm de ser facilmente integrados de uma forma orientada a serviços dentro dos serviços já existentes nas empresas. Em [35] os autores propõem uma integração baseada em serviços *Web* dos sistemas empresariais com as atividades do chão da fábrica, utilizando dispositivos com capacidades embebidas já prontas para a arquitetura SOA. São examinados os requisitos necessários à integração dos dispositivos para a obtenção de uma arquitetura apropriada que pretende eliminar a lacuna existente neste tipo de integração. O fornecimento em tempo real de informação, o impacto da informação dos dispositivos ao nível das aplicações de decisão assim como a comunicação bidirecional com o nível dos serviços dos dispositivos promove a visão de uma produção adaptativa que leva à redução dos custos de produção.

2.5 Planeamento e sequenciamento da produção

Os sistemas produtivos envolvem muitas decisões a serem tomadas a nível de planeamento, especialmente nos casos em que existe um elevado número de recursos a serem geridos de forma a garantir não apenas os prazos de entrega bem como a minimização dos custos do processo produtivo.

Com o elevado número de decisões a terem de ser tomadas, o planeamento é abordado de forma hierárquica dividindo-se em geral as decisões de acordo com os problemas que envolvem. Assim sendo, ao serem tomas decisões de um determinado nível hierárquico irão surgir restrições para os níveis inferiores.

A divisão hierárquica do planeamento é comum ser dividida em três níveis diferentes, de acordo com o seu horizonte temporal: planeamento estratégico ou planeamento de longo prazo, planeamento tático ou planeamento de médio prazo e planeamento operacional ou planeamento a curto prazo.

O planeamento estratégico define as estratégias a serem seguidas pela empresa, de modo a torna-la competitiva e obter o sucesso necessário para sobreviver no mercado competitivo. As decisões mais importantes tomadas a este nível introduzem as maiores alterações ao nível estrutural e provocam efeitos a longo prazo de onde se destacam [36]:

- definição dos objetivos a cumprir;
- sistemas de produção e distribuição - escolha, alteração ou implementação;
- instalações da empresa - localização e dimensionamento;
- produtos - desenvolvimento de novos produtos;
- logística - possibilidade de introdução de novos sistemas;
- equipamentos - estudo de aquisição de novos equipamentos.

Ao nível intermédio, encontram-se as decisões a médio prazo que entram no domínio do planeamento tático. A este nível os gestores concentram-se em tomar decisões sobre a utilização dos recursos produtivos tais como a alocação das capacidades necessárias de cada família de produtos, a utilização dos recursos de armazenamento e de distribuição através da criação de stocks ou alternativas de transporte dos produtos, e sobre a mão-de-obra disponível, onde se analisa a necessidade de introdução de novos recursos, horas

de trabalho extra ou mesmo a subcontratação para a conclusão dos trabalhos no tempo devido [36].

Por fim, nas decisões a curto prazo é necessário separar toda a informação e decisões provenientes dos níveis superiores. A este nível, as decisões mais comuns centram-se em [36]:

- alocação de tarefas para os recursos, através da definição da sequencia de diferentes atividades a serem realizadas;
- dimensionamento de lotes de artigos e volumes de produção;
- gestão de stocks;
- gestão de frotas.

É neste último nível de planeamento que é definido como tudo é produzido, por quem, onde, com que recursos, o que leva a que seja o nível onde existe um maior detalhe contrariamente ao que acontece em níveis superiores.

Como foi referido, na pirâmide hierárquica as decisões de sequenciamento correspondem ao planeamento operacional e são o último passo no processo de transformação antes de ocorrer qualquer tipo de produção.

Segundo [37], o objetivo do sequenciamento é "garantir que as tarefas adequadas são executadas no momento exato e no recurso certo". Com isto, o sequenciamento da produção consiste num processo de decisão que procura fazer uso eficiente dos recursos de produção, com incidência predominante nos meios de produção e assegurar a rápida execução dos trabalhos, sempre com o objetivo de otimizar um ou mais critérios.

Suponha-se que existem n trabalhos $J_i(i = 1, \cdots, n)$ que têm de ser processados em m máquinas $M_i(i = 1, \cdots, m)$. Por sua vez, cada trabalho pode incluir variadas tarefas.

Tipicamente existe um determinado número de tarefas à espera de serem processadas. O sequenciamento preocupa-se em determinar a ordem de processamento dos trabalhos J_i à entrada de uma máquina. As regras baseadas em prioridades são heurísticas simples usadas para determinar a ordem pela qual as tarefas deverão ser processadas. Algumas das regras mais utilizadas podem ser encontras na tabela 2.1.

As regras de prioridades podem ser classificadas como locais ou globais. As regras locais consideram a informação relativa a apenas um centro de processamento enquanto que as regras globais tomam em consideração informação referente a múltiplos centros de processamento. As regras FCFS, SPT e EDD são regras locais e a regra CR global. As regras locais são particularmente úteis para operações do tipo gargalo.

Apesar de ser geralmente reconhecida a elevada importância do processo de sequenciamento na atividade produtiva de qualquer organização, a abordagem industrial ao sequenciamento tende a ser simplista, resultando por isso frequentemente em soluções de qualidade modesta [37].

A razão de tal abordagem simplista resulta, aparentemente, por um lado da falta de conhecimento da existência de métodos de qualidade que podem oferecer melhores soluções e, por outro, da dificuldade, conhecendo-os, de os implementar e utilizar na prática. Isto é verdade principalmente quando os métodos necessitam de ser implementados em, ou integrados com sistemas computacionais de apoio ao sequenciamento da produção associados a sistemas ERP ou sistemas MES para poderem ser utilizados pelas empresas. Esta dificuldade é contornada recorrendo predominantemente a implementações de

Tabela 2.1: Principais regras de prioridade [38]

Regra de prioridade	Descrição
First Come, First Served (FCFS)	As tarefas são produzidas pela ordem em que chegam ao processador
Shortest Processing Time (SPT)	As tarefas são sequenciadas pela duração do seu tempo de processamento na máquina, do menor para o maior
Longest Processing Time (LPT)	Semelhante à regra anterior porém as tarefas são selecionadas do maior para o menor tempo de processamento
Earliest Due Date (EDD)	As tarefas são selecionadas com base no prazo de entrega mais próximo
Critical Ratio (CR)	As tarefas são processadas de acordo com o menor rácio ente o tempo restante até à data de entrega e o tempo de processamento dessa tarefa

mecanismos simples baseados em regras de prioridades de execução aos trabalhos, como é o exemplo a prioridade baseada na urgência dos trabalhos em relação às suas datas de entrega acordadas ou em mecanismos heurísticos simples [39].

A eficácia de qualquer sequenciamento de tarefas pode ser julgada em termos de um ou mais critérios de desempenho. Os medidores que são usados com mais frequência são [38]:

- Tempo de fluxo - mede o tempo que uma tarefa passa no sistema. A minimização deste valor é apropriada quando se pretende uma rotação rápida dos produtos no sistema e a manutenção de baixos níveis de existências;

- Tempo de atraso - este critério torna-se útil quando a empresa inclui uma penalização por unidade de tempo se a conclusão de uma tarefa se atrasa para além do prazo de conclusão estipulado;

- *Makespan* - mede o tempo total para a conclusão de um grupo de tarefas, ou seja, mede o tempo entre o inicio da primeira tarefa de um grupo até à conclusão da tarefa desse mesmo grupo. No caso de uma única máquina, o *makespan* será sempre o mesmo independentemente da regra de prioridade utilizada.

O planeamento e sequenciamento podem ser atividades difíceis por um variado numero de razões. Uma é que na realidade uma operação precisa lidar com variações nos tempos de preparação, tempos de processamento, interrupções modificações no conjunto de tarefas a serem produzidas. Outra razão reside no facto de não haver nenhum método que identifique um planeamento ou sequenciamento ótimo sendo virtualmente impossível a organização ente um vasto número de possibilidades de alternativas para a obtenção do valor ótimo. Porém, a introdução das TI veio trazer novas possibilidades no planeamento tornando a execução em tempo real uma possibilidade.

2.5.1 Nomenclatura e representação

Existe uma grande diversidade de problemas de sequenciamento, associados normalmente a problemas de produção. Esta diversidade levou ao estabelecimento de notações específicas para os problemas de sequenciamento e que pode variar de autor para autor.

Para este trabalho foi adotada a notação presente em [37] que descreve um problema de sequenciamento através de três variáveis segundo a forma $\alpha \mid \beta \mid \gamma$. O campo α contém apenas uma entrada e descreve o ambiente de processamento. O campo β fornece detalhes acerca das características e restrições. Pode conter um parâmetro, multi-parâmetros ou nenhum parâmetro. Finalmente o campo γ descreve o objetivo a ser minimizado e, na maioria dos casos, contém apenas um critério.

Os ambientes de processamento que se podem encontrar com mais frequência no campo α, que indicam o tipo de problema são [37]:

- máquina única (1) - O caso de uma única máquina corresponde ao ambiente de processamento mais simples que se pode encontrar;
- Máquinas idênticas em paralelo (Pm) - Existem m máquinas idênticas organizadas em paralelo. Um trabalho j que requer uma única operação pode ser executado em qualquer uma das m máquinas existentes;
- *Flow-Shop* (Fm) - neste ambiente existem m máquinas em serie. Cada tarefa tem de ser processada em cada uma das m máquinas e todos os trabalhos têm de seguir o mesmo trajeto.

O campo β, que indica as características da tarefa (independente/ restrições de precedência), por omissão (se o campo β estiver vazio) assume-se que todas as tarefas estão disponíveis para processamento ao mesmo tempo, não é permitida a interrupção e não existem restrições de precedência. Podem ainda ser encontradas um grande número de restrições, sendo as mais importantes:

- Data de lançamento (r_j) - se esta restrição aparecer no campo β então, o trabalho j não pode ser processado antes da sua data de lançamento r_j. Se r_j não aparecer no campo β, o processamento do trabalho pode ser iniciado a qualquer momento;
- Precedência (*prec*) - esta restrição pode aparecer em ambientes de máquina única ou de máquinas em paralelo e requer que um ou mais trabalhos sejam concluídos antes de outro conjunto de trabalhos receber permissão para ser processado;
- Avarias (*brkdwn*) - as avarias da máquina implicam que este não poderá estar continuamente disponível.
- Tempos de preparação dependentes da sequência (s_{jk}) - representa os tempos de preparação que ocorrem entre o processamento dos trabalhos j e k. Este tempos dependem da máquina que for considerada;

Para o caso do campo γ, onde está identificada a função objetivo, os problemas encontrados com mais frequência são [37]:

- Minimização do *makespan* (C_{max}) - minorar o tempo que se demora a concluir um determinado número de tarefas. Um *makespan* baixo implica uma boa utilização da(s) máquina(s);

- Minimização de atrasos (T_{max}) - tenta minimizar, quanto possível, minimizar o número de tarefas em violação dos prazos de entrega;

- Minimização do tempo total de conclusão com pesos ($\sum w_j C_j$) - procura minimizar a soma de todos os tempos de conclusão em que são atribuídos pesos às tarefas. Fornece indicações dos custos de inventários que incorrem do planeamento.

- Minimização do total de atrasos com pesos ($\sum w_j T_j$) - procura minimizar o tempo total da violação de prazos de entrega tendo cada uma das tarefas um peso associado em função de prioridades estabelecidas.

2.5.2 Problemas de máquina única

No universo de problemas relacionados com o sequenciamento, os problemas de máquina única são os problemas mais estudados na literatura. Nestes problemas, considera-se, em geral, que todas as tarefas podem iniciar a sua produção no instante inicial ($t = 0$) e que cada uma das tarefas tem um tempo de processamento e prazo de entrega bem definidos [36].

Um dos casos mais comuns onde este tipo de problemas está presente é, por exemplo, quando se pretende minimizar o atraso máximo, considerando n tarefas, cada uma, j com um tempo de processamento p_j, e um prazo de entrega, d_j bem definidos. Todas as tarefas são processadas numa única máquina que só pode realizar uma tarefa de cada vez.

2.5.3 Problemas de máquinas paralelas

No caso de estarem disponíveis recursos com funcionalidades idênticas, esta-se perante um problema de máquinas paralelas, muitas vezes de elevada complexidade uma vez que se torna necessário para além do sequenciamento de tarefas a atribuição das tarefas aos recursos existentes.

Estes problemas são caracterizados por possuírem m máquinas e n tarefas a serem processadas, cada uma com um tempo de processamento p_j. As tarefas só podem ser processadas numa única máquina e cada máquina só poderá processar uma tarefa de cada vez. Assume-se ainda que a máquina i processa a tarefa j com velocidade s_{ij}.

Os problemas de máquinas paralelas podem ainda ser divididos em três grandes classes [36]:

- Caso de máquinas idênticas em que o processamento de uma tarefa é igual em todas as máquinas, isto é, as máquinas têm a mesma velocidade de processamento;

- Caso de máquinas uniformes: onde cada máquina i tem associada uma velocidade de processamento;

- Caso de máquinas não relacionadas onde não há nenhuma relação entre as velocidades de processamento, isto é, os s_{ij} são específicos das tarefas e das máquinas.

2.5.4 Complexidade dos problemas de planeamento

Os problemas de planeamento são problemas de otimização, ou seja, problemas cujo objetivo consiste em chegar a uma solução tendo em vista a minimização e/ou maximização de um ou mais critérios de otimização. Porém, dependendo do tipo de problema, estes podem representar níveis de complexidade diferentes. Estes problemas podem ser de classe P (polinomiais) ou NP (não polinomiais).

A teoria da complexidade computacional, responsável pela classificação dos problemas computacionais de acordo com a sua dificuldade inerente, define a classe de complexidade P é muitas vezes vista como uma abstração matemática de um problema cujas soluções podem ser obtidas em tempo polinomial, e consequentemente permite a utilização de algoritmos exatos. Por outro lado, a classe de complexidade NP diz respeito a problemas em que não é conhecido nenhum algoritmo exato para a obtenção de soluções [40].

Um algoritmo de aproximação, ou não exato, procura encontrar a solução mais próxima do valor ótimo. Estes algoritmos substituem a exatidão da solução à custa de um menor tempo de processamento. Alguns exemplos de algoritmos mais conhecidos são: algoritmos genéticos que representam uma classe de algoritmos evolutivos baseado na teoria da evolução utilizado para encontrar soluções aproximadas para problemas de otimização; arrefecimento simulado que é uma meta-heurística de otimização através de uma técnica de busca local probabilística, baseada nos fundamentos da termodinâmica, *branch-and-bound* que consiste numa enumeração sistemática de todos os candidatos a soluções para encontrar respostas a problemas de otimização, entre muitos outros.

Capítulo 3

Caso de estudo

A empresa onde se incidiu o estudo foi a GNS. Esta empresa encontra-se localizada em Águeda, mais especificamente na EN1 - Mourisca do Vouga. É uma empresa industrial que conta com mais de 50 anos de experiência na produção de fechaduras e acessórios para caixilharia metálica, seja em perfis de alumínio, ferro ou aço inoxidável.

Dispõem de umas instalações fabris ocupando uma área aproximada de $7000m^2$ onde trabalham 70 colaboradores e uma faturação aproximada de 5.000.000€ dos quais 50% procedem da exportação.

No que diz respeito a produtos, podem ser encontradas fechaduras Monoponto e Multiponto, Muletas, Cremones, acessórios de fecho e de fixação e sistemas de Oscilo Batente com ou sem micro ventilação, tudo o que é necessário para o fabrico de portas e janelas em perfil de alumínio. Um exemplo de fechadura pode ser visto na figura 3.1.

Figura 3.1: Fechadura GNS 7715

3.1 Processo produtivo

3.1.1 Analise A-B-C

A análise A-B-C classifica os inventários de acordo com uma determinada medida de importância, normalmente de valor monetário (faturação total anual) por forma a diferenciar esforços de acordo com a sua importância. Tipicamente são utilizadas três categorias: categoria A (importância elevada), categoria B (importância moderada) e categoria C (importância menor). Porém, o número de categorias pode variar de orga-

nização para organização dependendo do objetivo desta na extensão de diferenciamento de esforços [38].

Através da análise A-B-C a empresa determinou qual ou quais os produtos responsáveis por 75% da faturação total anual e passou a dar uma maior prioridade à fabricação destes produtos passando os restantes para um segundo plano. Assim, assegurando que os produtos da categoria A são sempre satisfeitos está assegurado 75% do rendimento anual da empresa. Os produtos que se encontram incluídos dentro desta categoria são fechaduras com as referências 7715 e 7716.

3.1.2 Análise da Produção

Todo o processo produtivo da empresa assenta no conceito da produção do tipo Pull. Este sistema regista as necessidades junto do consumidor final, que no caso desta empresa pode ser o seu armazém ou uma ordem específica, para determinar as quantidades de materiais a movimentar entre os diferentes estados de produção (movimentar/produzir apenas aquilo que é necessário na quantidade requerida e no tempo adequado).

Juntamente com o sistema Pull, também é adotado um sistema de cores (vermelho, amarelo e verde) para a gestão dos stocks entre os postos de trabalho. Se determinado stock de peças se encontrar no verde significa que garante a produção para pelo menos 4 semanas, amarelo para 2 semanas e vermelho para 1 semana. Assim que o stock desce abaixo do verde é lançado um alarme para a reposição desse stock e, caso chegue a vermelho, a reposição do stock é forçada na linha. De certa forma, o sistema de cores pode ser associado a um sistema do tipo Kanban que é um sistema para controlar o fluxo de peças e materiais entre os vários postos de trabalho. Este sistema foi desenvolvido por Taiichi Ohno, na Toyota [38], de forma a encontrar um sistema que permitisse melhorar e manter uma produção elevada alcançado o JIT. O controlo de peças e materiais em movimento respondem a sinais de necessidades que podem ser lançados através de cartões, sinais luminosos ou sistema de cores como no caso da empresa em caso de estudo.

Assim que um componente entra em falta no devido armazém intermédio, ou seja, aquando do lançamento do alarme para a reposição desse componente, a ordem para a linha de fabricação ou compra é lançada. Contudo, acontece que são lançadas ordens de reposição para vários componentes em simultâneo pelo que é necessário um critério para a seleção de quais as ordens a serem satisfeitas em primeiro e depois as restantes em segundo lugar. No caso da GNS esta diferenciação é feita através de um sistema de prioridades que advém da análise A-B-C efetuada anteriormente. Esta ordem de prioridades, para os produtos da categoria A é apresentada de seguida:

1. 7715 x 35 T/L 1,60 ECO(T 22)
2. 7716 x 35 T/L
3. 7715/A x 35 T/L 1,60 (T 22)
4. 7716 x 35/A T/L
5. 7715 x 35 T/L 1,60
6. 7715 x 35 T/L 1,80

Na figura 3.2 um exemplo de como é que esta lista de prioridades é utilizada dentro da empresa.

Das ordens de produção que se encontram a vermelho, ou seja, aquelas que tem prioridade sobre as restantes, pode-se verificar que contém uma lista de peças necessárias para a reposição de stocks. Cruzando a informação com a lista de prioridades implementada pela empresa, verifica-se que em primeiro lugar se produzem os componentes para a fechadura 7715 e depois os componentes para a fechadura 7716. Após a conclusão dos componentes encontrados na lista de prioridades os restantes são fabricados conforma a ordem em que aparecem.

No caso deste exemplo as peças para os produtos 7715 e 7716 aparecem no topo da lista, mas no caso de aparecerem no fim da lista e conforme a lista de prioridades, estes têm de ser produzidos em primeiro e só depois os restantes componentes.

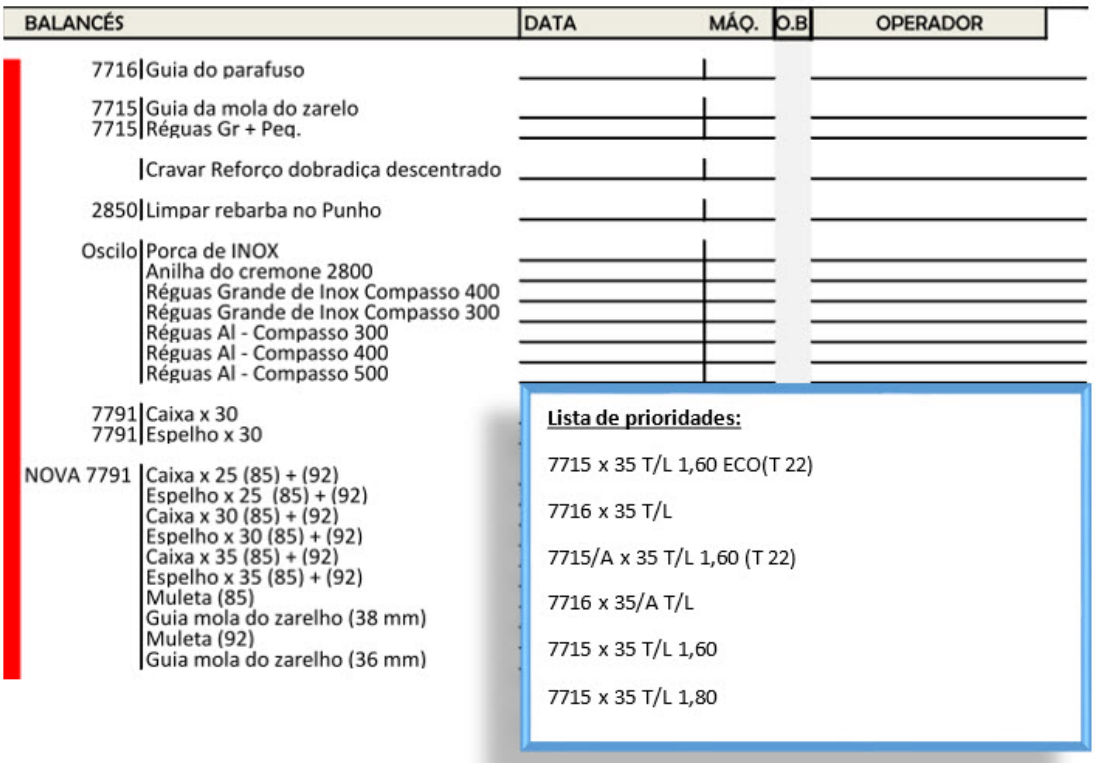


Figura 3.2: Exemplo da utilização das prioridades nos balancés

3.2 Ambiente de produção da empresa

O ambiente de produção da empresa GNS encaixa-se no tipo de problema de escalonamento de uma única máquina. Este é um sistema de produção constituído por um processador único, disponível para executar os trabalhos.

Num sistema de processador único existe uma fila de tarefas em espera, havendo apenas um tipo de decisão a tomar, que passa pela determinação da ordem de execução dos trabalhos no dado processador (figura 3.3).

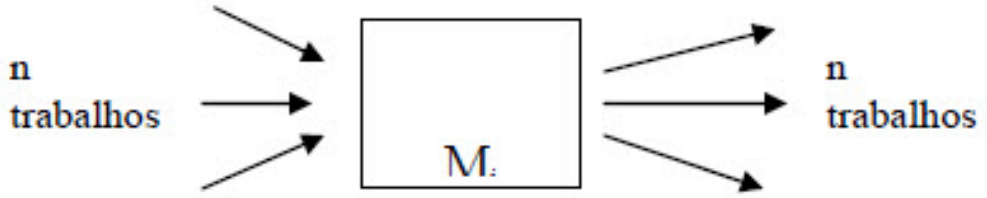


Figura 3.3: Sistema de processador único

De acordo com a classificação de três campos $\alpha\ |\beta\ |\gamma$ proposta por [37] o problema pode ser definido por $1\ |\ |\sum w_j \cdot T_j$ que traduz um ambiente de processador único, sem restrições onde o objetivo passa por minimizar a totalidade dos trabalhos atrasados com pesos atribuídos em função da sua prioridade.

Considere-se que existem n tarefas disponíveis no instante inicial para serem processadas onde existe apenas um processador. Cada uma das tarefas tem associada um peso positivo w_j e uma data de entrega d_j. Ou seja, cada tarefa j precisa de uma única operação cujo tempo de processamento p_j é conhecido. No caso da tarefa ser terminada depois da data de entrega d, este atraso é dado por $T_j = C_j - d_j$, onde C_j corresponde ao lugar no tempo em que a tarefa é iniciada. A interrupção de uma tarefa durante o seu processamento não é permitida e no instante de início do processamento, no instante 0, todas as tarefas encontram-se disponíveis.

Dado uma lista de tarefas para escalonar, o objetivo passa pela minimização da função 3.1.

$$Min\ z = \sum_{j=1}^{n} w_j \cdot T_j \tag{3.1}$$

No ambiente moderno de competitividade entre as empresas , os custos associados a entregas tardias podem resultar na perda de interesse por parte dos clientes, aumento nos custos em envios urgentes não planeados e consequente perda de lucros. Desta forma, a minimização dos atrasos de tarefas com pesos atribuídos não se trata apenas de uma formulação de interesse académico como de bastante interesse na aplicação prática.

Este problema foi provado em [41] como sendo altamente NP-difícil, significando que não consegue ser resolvido pelos métodos tradicionais de cálculo e tem recebido bastante atenção devido à sua complexidade computacional.

3.2.1 Alguns trabalhos desenvolvidos na resolução do problema

Na literatura têm sido estudados diferentes métodos para a resolução do problema dos trabalhos atrasados com pesos atribuídos, representado pela equação 3.1. Em [42] é desenvolvido um algoritmo determinístico baseado na pesquisa tabu para o problema da totalidade dos trabalhos atrasados com pesos atribuídos. A pesquisa tabu trata-se de um procedimento adaptativo auxiliar, que guia um algoritmo de busca local na exploração

contínua dentro de um espaço de busca. Este método permite a resolução inteligente de problemas através da incorporação de memória adaptativa e exploração responsiva a alterações. Os elementos chave do caminho de pesquisa são seletivamente relembrados e são executadas escolhas estratégicas para guiar a pesquisa para fora da zona do ótimo local e assim fazer uma pesquisa de um espaço de soluções mais alargado. O método, a partir de uma solução inicial, move-se, de forma iterativa, da solução corrente para o melhor vizinho, mesmo que a solução desse vizinho seja pior que a solução corrente. Este processo é executado até ser verificado o critério de paragem. Para evitar que o algoritmo fique preso em ótimos locais, determinados movimentos são proibidos e classificados como tabu.

Em [43] são utilizadas duas meta-heurísticas, a otimização por enxame de partículas e algoritmos evolutivos diferenciais, para a resolução do problema da totalidade dos trabalhos atrasados com pesos em uma máquina única.

A otimização por enxame de partículas é um método de cálculo que otimiza um problema iterativamente ao tentar melhorar uma solução candidata no que diz respeito a uma dada medida de qualidade. O método otimiza um problema através de uma população de soluções candidatas (partículas), movendo-se em torno dessas partículas no espaço de busca de acordo com fórmulas matemáticas simples sobre a posição e velocidade da partícula. O movimento de cada partícula é influenciado pela sua posição local mais conhecida, mas, também é orientado em direção às posições mais conhecidas no espaço de busca, que são atualizadas à medida que são encontrados por outras partículas. Desta forma o enxame move-se para as zonas de melhores soluções. No caso do algoritmo evolutivo diferencial, a população alvo é perturbada pelo operador de mutação e o operador de cruzamento que introduz a combinação entre a população mutada e a população alvo, de modo a gerar uma população de ensaio. Em seguida, o operador de seleção é aplicado para comparar o valor da função de aptidão de ambas as populações concorrentes. No final, os melhores indivíduos tornam-se membros da população que passa para a próxima geração. Este processo é repetido até que seja verificado o critério de paragem.

Os resultados computacionais obtidos, mostraram que os dois métodos utilizados devolvem soluções promissoras para os problemas de natureza combinatória. Para aumentar a qualidade da solução, foi ainda implementado um método de pesquisa baseado na pesquisa de vizinhança variável em ambos os algoritmos. Comparativamente os melhores resultados são obtidos pelos métodos que fazem uso da pesquisa por vizinhança variável. Também foi observado que para o mesmo tempo de processamento o algoritmo de evolução diferencial obtém soluções mais rápidas que o método por enxame de partículas, o que pode ser mais indicado para problemas de maior dimensão.

Em [44] é validado um algoritmo baseado na colónia de formigas com problemas de *benchmark* encontrados na literatura da área da investigação operacional. Este método é uma heurística baseada em probabilidade, criada para solução de problemas computacionais que envolvem procura de caminhos em grafos. Este algoritmo foi inspirado na observação do comportamento das formigas ao saírem de sua colónia para encontrar comida. Os resultados obtidos foram comparados com o melhor resultados disponíveis que mostraram ser o mais perto do ideal. Os resultados computacionais obtidos permitiram concluir sobre sua eficiência e eficácia.

No caso de [45] é apresentado um algoritmo híbrido para a resolução de problemas de otimização de natureza combinatória. O algoritmo desenvolvido faz a combinação entre a técnicas de evolução diferencial com uma pesquisa de vizinhança variável. A

evolução diferencial é usada como um otimizador global de forma a guiar a evolução da população de soluções para as regiões ótimas do espaço de soluções enquanto que a técnica da procura por vizinhança variável é utilizada como um otimizador local, executando alterações locais nos indivíduos da evolução diferencial até ser encontrado um ótimo local. Os resultados mostraram que podem ser obtidas soluções de boa qualidade em tempos reduzidos para o problema das tarefas atrasadas com pesos.

Em [46] foi abordado o problema da minimização dos trabalhos tardios com pesos do ponto de vista da existência de uma data comum a todas as tarefas. Os autores desenvolvem um algoritmo através da programação dinâmica como alternativa aos algoritmos de aproximação polinomial usados em trabalhos, cujos resultados obtidos mostraram-se encorajadores.

Bons resultados também foram obtidos em [47] através da utilização de algoritmos genéticos.

Contrariamente aos métodos de otimização de natureza não exata, como os que foram implementados pelos autores referenciados anteriormente, em [48] e em [49], é proposto o uso do algoritmo *branch-and-bound*, um algoritmo de natureza exata. A implementação paralela do algoritmo permitiu a redução significativa do tempo computacional e resolver problemas de dimensões maiores. O algoritmo foi testado para um conjunto de problemas de teste e obteve resultados de grande precisão para problemas de tamanho superior a 50 tarefas com tempos de processamento significativamente menores aos encontrados até ao momento para este algoritmo.

Capítulo 4

Métodos matemáticos e tecnologias de suporte

4.1 Algoritmos genéticos

Em 1859 Charles Darwin propôs a teoria da evolução na sua obra *"On the Origin of Species"* onde explica as suas observações de plantas e animais no seu habitat natural. Darwin observou que os indivíduos dentro de um ecossistema competem entre si por recursos limitados e os menos aptos tendem a morrer numa competição por comida ou água, e que os sobreviventes levam ao melhoramento da espécie. O conceito de seleção natural foi utilizado para explicar como as espécies têm sido capazes de se adequarem às mudanças de ambiente como, consequentemente, espécies que são muito idênticas em adaptabilidade conseguem também elas evoluir. A combinação das características dos indivíduos que sobrevivem podem levar ao aparecimento de um novo indivíduo melhor adaptado às características do seu meio ambiente ao adquirir características positivas de cada um dos progenitores [50].

Os Algoritmos Genéticos (AGs) foram apresentados em 1975 pelo Professor John Holland na Universidade de Michigan, no mesmo ano da sua publicação *"Adaptation in Natural and Artificial Systems"*. David Goldberg, aluno de Holland, obteve o primeiro sucesso em aplicação industrial com AGs em 1989 aquando da publicação da sua obra *"Genetic Algorithms in Search, Optimization, and Machine Learning"*. Ou seja, estes algoritmos consistem em procedimentos computacionais que imitam os processos naturais de sobrevivência e reprodução das populações numa evolução constante [51].

Estes algoritmos pertencem às técnicas de procura local mas com algumas particularidades. Na figura 4.1 pode ser observada a estrutura básica deste tipo de algoritmos. A principal característica é que em vez de se iniciar o processo com uma solução inicial, o processo é iniciado com um conjunto inicial de soluções possíveis. Os AGs quando aplicados à programação da produção vêm as soluções como indivíduos ou membros de uma população. Cada indivíduo é avaliado segundo uma função de aptidão que ditará a sua sobrevivência na população. Para cada nova geração novos indivíduos são selecionados para reprodução. Os escolhidos são submetidos ao operador cruzamento (*crossover*) para gerar novos indivíduos. Os recém nascidos podem, de seguida ser sujeitos ao operador de mutação com uma determinada probabilidade de mutação. Com a evolução das gerações os novos indivíduos podem substituir os antigos na totalidade da população.

Cada indivíduo na população é denominado de *string* ou cromossoma, analogamente ao sistema natural. Normalmente estes indivíduos são codificados em cromossomas binários em que cada elementos que forma o cromossoma é designado de gene. De forma resumida, os AGs trabalham com uma população de indivíduos, cada um com o seu cromossoma característico, onde estão os genes que o identificam, podendo ser mais ou menos apto para determinado problema. O tamanho da população determina a quantidade de informação guardada pelo AG.

Para a utilização de um AG a codificação da informação em cromossomas e a definição de uma função de aptidão, tornam-se as tarefas mais importantes.

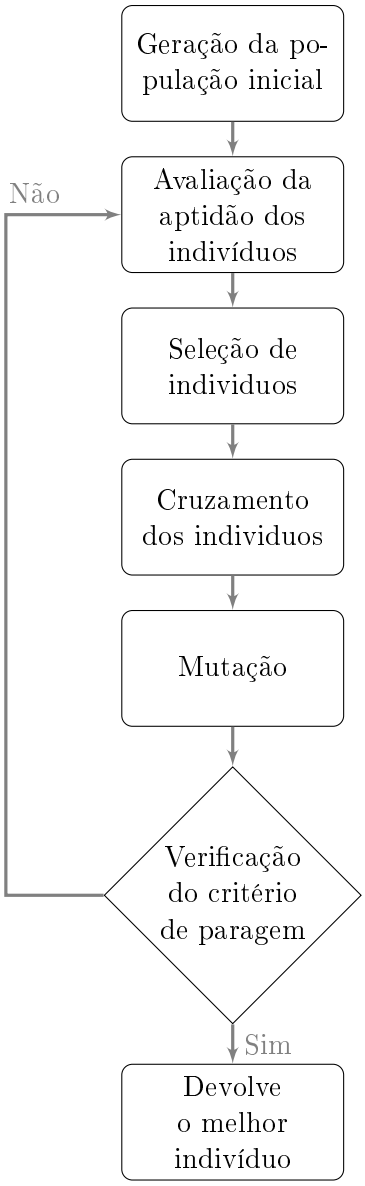


Figura 4.1: Estrutura básica de um algoritmo genético [52]

4.1.1 Função de aptidão

A função de aptidão é utilizada pelo AG para avaliar a qualidade que cada indivíduo da população detém para a solução do problema. Esta será a componente mais importante de qualquer AG. É através desta função que se mede o quão próximo um indivíduo está da solução desejada e quão boa é a solução analisada. Torna-se essencial que esta função seja representativa e diferencie na proporção correta as boas e as menos boas soluções. No caso de pouco precisão na avaliação corre-se o risco de uma solução ótima poder ser descartada durante a execução do algoritmo, além de gastar mais tempo na geração de um maior número de soluções menos promissoras[52].

4.1.2 Seleção

Podem ser escolhidos vários métodos para selecionar os indivíduos sobre os quais serão aplicados os operadores genéticos, como por exemplo, seleção por roleta e seleção por torneio.

O método da roleta é uma forma de seleção proporcional de cada indivíduo da população. A probabilidade de escolha de um indivíduo é proporcional ao seu índice de aptidão. Desta forma, aos indivíduos com maior aptidão é dada uma porção maior da roleta enquanto que a parte menor é dada aos indivíduos com menor aptidão. Um possível esquema de método da roleta está ilustrado na figura 4.2.

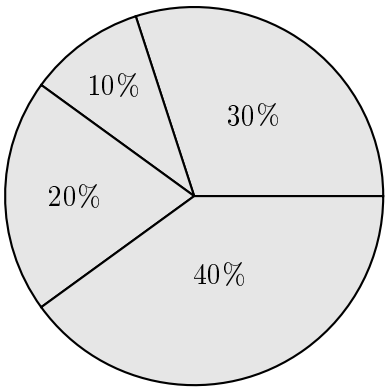


Figura 4.2: Método de seleção por roleta(baseado em [52])

Na seleção por torneio, n indivíduos são selecionados aleatoriamente formando um grupo ou sub-população sendo selecionado o melhor indivíduo desse grupo.

O objetivo dos AGs é convergir para uma solução ótima, e a pressão de seleção *(selection pressure)* é a força condutora que determina a taxa de convergência. Uma alta pressão de seleção leva a uma convergência mais rápida da população. A seleção por roleta fornece pressões de seleção mais elevadas na geração inicial, especialmente quando existem poucos indivíduos que têm valores de aptidão significativamente mais altos que outros. A seleção por torneio fornece mais pressão de seleção nas gerações seguintes, quando os valores de aptidão dos indivíduos não apresentam diferenças muito significativas.

4.1.3 Cruzamento

Assim que dois cromossomas são selecionados, o operador cruzamento pode ser usado para gerar dois descendentes. O cruzamento dos genes pode ser feito num único ponto, denominado de cruzamento de ponto único, ou em dois ou mais pontos, sendo assim designado de cruzamento multi ponto. A escolha do ponto de cruzamento é feita de forma aleatória mas limitada ao comprimento do cromossoma, ou seja, a posição de cruzamento está entre 1 e $(L-1)$, em que L representa o comprimento do cromossoma. Por exemplo, num cruzamento de ponto único (figura 4.3), o primeiro descendente contém os genes do primeiro progenitor até ao ponto de cruzamento e a segunda parte dos genes do segundo progenitor após o ponto de cruzamento.

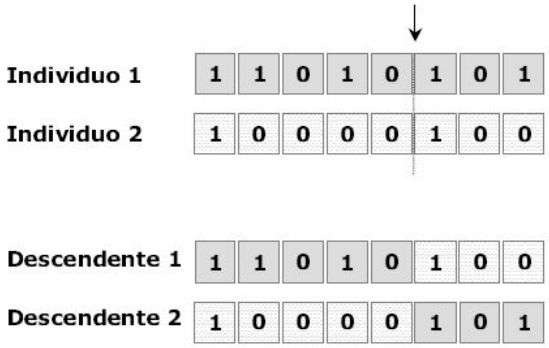


Figura 4.3: Exemplo de um cruzamento de pronto único [52]

O operador cruzamento utilizado tem uma grande influência no desempenho de um AG. O número de operações de cruzamento é controlado pela probabilidade de cruzamento, normalmente definida pelo rácio entre os descendentes gerados a cada nova geração e a população total. Uma elevada probabilidade de cruzamento permite explorar um maior espaço de soluções e reduz as hipóteses de obter um ótimo local, enquanto que uma probabilidade baixa permite explorar os indivíduos existentes na população que tenham valores de aptidão mais elevados.

4.1.4 Mutação

No caso de cromossomas binários, a mutação pode ocorrer através da inversão de um *bit* (figura 4.4), enquanto que num cromossoma não binário, a mutação pode ocorrer através da permuta de genes dentro do cromossoma, de forma aleatória.

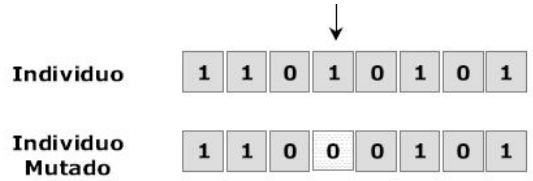


Figura 4.4: Operador de mutação de ponto único [52]

O operador de mutação mostra-se necessário para a introdução e manutenção da

diversidade de material genético da população, permitindo ao algoritmo chegar a qualquer ponto do espaço de soluções. Normalmente este operador é aplicado através de uma probabilidade de mutação geralmente pequena.

A probabilidade de mutação é definida como a probabilidade de frequência de mutação de cada gene. Isto controla a taxa a que um novo gene é introduzido na população. Se este for demasiado baixo, muito genes que são úteis nunca serão descartados. Se por outro lado for muito elevado ira ocorrer uma perturbação exagerada com os descendentes a perderem as semelhanças com os criadores. Dessa forma, a habilidade do algoritmo em aprender com os mais "velhos"irá ser perdida.

À semelhança do operador de cruzamento, a mutação pode ser implementada num único ponto, sendo designada de mutação de ponto único, ou em várias posições , mutação multi-ponto. A escolha da posição de mutação é sempre aleatória.

4.1.5 Critério de paragem

Assim que o AG é inicializado, entra num processo ciclicamente evolutivo e torna-se necessário estabelecer uma meta, não ideal, para que este não seja executado de forma infinita. O critério principal é a deteção de uma solução ótima, mas como este cenário pode por vezes ser hipotético, outro critério utilizado passa pela limitação do tempo de execução. No entanto, este pode comprometer, em casos em que a taxa de evolução seja reduzida, a descoberta de uma boa solução. Limitar o número de gerações é também outro método utilizado, bem como definir o valor de aptidão mínimo, médio ou máximo. Existem ainda critérios fundamentados no próprio acompanhamento do processo evolutivo, ou seja, sempre que for detetada evolução nas aptidões dos indivíduos da população, o processo evolutivo continua. Caso não haja melhoria nas últimas gerações o processo converge e termina [52].

4.2 SOAP

O protocolo de comunicação *Simple Object Acess Protocol* (SOAP) consiste num protocolo baseado em XML direcionado para a troca de mensagens estruturadas entre computadores em ambiente descentralizado e distribuído [53], ou seja, permite a fácil comunicação entre aplicação cliente e aplicação servidor onde estão alojados os serviços *Web* à espera da mensagem para serem invocados, figura 4.5.

Atualmente, podem ser encontrados outros protocolos de comunicação incluindo CORBA, DCOM ou Java RMI porém, ao contrário destes protocolos, as mensagens em SOAP são escritas inteiramente em XML que as torna independentes da plataforma utilizada por parte das aplicações[54]. Desta forma, o protocolo SOAP mostra-se como um dos aspetos mais importantes na utilização de arquiteturas com serviços *Web* facilitando a troca de informações e dados ente diversas aplicações.

Este protocolo foi desenvolvido com o objetivo de proporcionar uma comunicação simples e leve. Ao efetuar uma chamada de um método remoto, o cliente não precisa de se preocupar com a alocação de memória pois o cliente executa o método do lado do servidor, semelhante a um processo local.

Uma mensagem SOAP é composta por três partes, um envelope que define toda a estrutura que descreve o conteúdo da mensagem e como esta vai ser processada, um

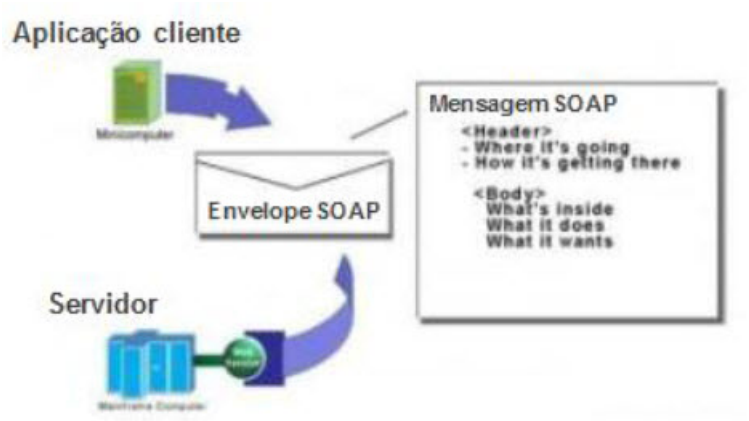


Figura 4.5: Utilização do SOAP pelos serviços *Web* [5]

conjunto de regras de codificação que representam as instâncias dos tipos de dados utilizados na aplicação e um mecanismo que define as invocações e respostas através de procedimentos remotos RPC [53].

Cada mensagem tem obrigatoriamente de incluir o elemento *Envelope*, o elemento opcional *Header* e uma elemento obrigatório *Body* (figura 4.6). Cada um destes elementos possui um conjunto de regras associadas.

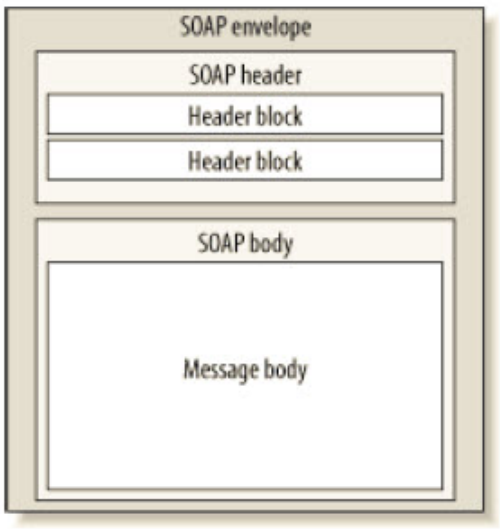


Figura 4.6: Estrutura de uma mensagem SOAP [55]

O elemento *Envelope* consiste no elemento raiz de uma mensagem SOAP. Codifica o documento XML como uma mensagem SOAP através da definição da etiqueta $< soap : Envelope >$. Também pode incluir atributos adicionais como o *encodingStyle*, responsável por definir o tipo de dados no documento XML.

No caso do elemento *Header*, trata-se de um elemento opcional e que contém informações específicas da mensagem SOAP, tais como autenticação, transações e encriptação, cuja etiqueta é definida por $< soap : Header >$.

No elemento *Body* é onde está definida a informação da mensagem SOAP. É definido pela etiqueta $< soap : Body >$ e pode ainda conter os métodos, parâmetros ou respostas do serviço *Web*. Na ocorrência de erros de processamento da mensagem SOAP, recorre-se ao sub-elemento *fault*, definido pela etiqueta $< soap : Fault >$. Contido neste elemento pode estar qualquer informação que possa ser expressa em sintaxe XML.

Na figura 4.7 é possivel observar um exemplo de um pedido SOAP. O pedido é apresentado na linha 7 onde é pedido a execução do método *getTemp*, um serviço *Web* que devolve a informação sobre o tempo que se faz sentir num determinado local através do seu código postal.

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <SOAP-ENV:Envelope
3          xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
4          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
6          <SOAP-ENV:Body>
7                  <ns1:getTemp
8                  xmlns:ns1="urn:xmethods-Temperature"
9                  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/
                       soap/encoding/">
10                         <zipcode xsi:type="xsd:string">10016</
                               zipcode>
11                 </ns1:getTemp>
12         </SOAP-ENV:Body>
13 </SOAP-ENV:Envelope>
```

Figura 4.7: Exemplo de um pedido SOAP [54]

O envelope SOAP contém o corpo da mensagem que contém um elemento que define a chamada remota de um procedimento, especificando o método pretendido (getTemp) com os argumentos (zipcode). O envelope é então transportado para um servidor SOAP através de um protocolo tal como HTTP ou SMTP. Após receber o pedido, o servidor invoca o método específico e gera a resposta que será transmitida à aplicação que a requereu. Na figura 5.7 é possível ver a mensagem SOAP com a resposta enviada à aplicação cliente com a temperatura para o código postal introduzido.

Após a receção da resposta do servidor, o cliente SOAP pode proceder à tradução da informação num formato mais amigável para o utilizador humano através da transformação do corpo da mensagem para um formato lido pelo utilizador humano como por exemplo HTML [56].

SOAP não se encontra restrito a nenhum protocolo de comunicação. De facto, as mensagens SOAP podem ser transportadas por qualquer protocolo de comunicação, sendo atualmente o protocolo HTTP o mais utilizado, no entanto é possível utilizar outros protocolos como por exemplo *File Transfer Protocol* (FTP), *Simple Mail Transfer Protocol* (SMTP) ou *Blocks Exchange Protocol* (BEEP).

Nos primórdios SOAP foi acrónimo para *"Simple Object Access Protocol"* e que mais tarde começou a ser utilizado como *"Service Oriented Architecture Protocol"* devido sua utilização frequente na arquitetura SOA. Contudo, na versão 1.2 da especificação do

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <SOAP-ENV:Body>
                <ns1:getTempResponse
                xmlns:ns1="urn:xmethods-Temperature"
                SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/
                    soap/encoding/">
                        <return xsi:type="xsd:float">71.0</return>
                </ns1:getTempResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 4.8: Exemplo resposta em SOAP [54]

W3C este deixou de ser considerado um acrónimo por levar a algumas confusões. Este protocolo teve o seu início no protocolo XML-RPC, uma especificação desenvolvida pela *Microsoft* em 1998, mais simples porém mais limitada que o SOAP. Atualmente, a especificação SOAP é mantida pela consórcio W3C [2].

Devido à grande adoção do protocolo como padrão respeitante ao serviços *Web* em aplicações de grandes organizações, tem levado ao desenvolvimento de SOAP *Application Programming Interface*(API)[5] para os produtos desenvolvidos facilitando em grande medida a integração de sistemas.

4.3 WSDL

O *Web Service Description Language* (WSDL) é uma especificação que descreve o serviço *Web* através da representação de um documentos XML, permitindo ao serviço *Web* ser auto-descritivo. De certa forma, WSDL representa um contrato entre o fornecedor do serviço e o cliente do serviço, da mesma forma que uma interface em linguagem Java representa um contrato com o código Java. A grande diferença reside na utilização de linguagem independente de plataforma para o caso do WSDL, através do uso de SOAP [54].

Através do WSDL um serviço *Web* descreve quais os métodos que disponibiliza e como os utilizar de forma a permitir a sua utilização por parte de uma aplicação cliente necessitando apenas de conhecer o documento com a sua descrição WSDL, normalmente disponibilizada através de um URL. A linguagem WSDL pode ser considerada como um manual de instruções, na medida em que descreve de que maneira se pode utilizar o serviço.

O uso de WSDL apresenta variadas vantagens das quais [55]:

[5]Uma API, ou Interface de Programação de Aplicativos é um conjunto de rotinas e padrões estabelecidos por um *software* para a utilização das suas funcionalidades por aplicativos que não pretendem envolve-se em detalhes da implementação do *software*, mas apenas usar os seus serviços.

- Facilita a escrita e manutenção de serviços através de uma abordagem estruturada na definição da interface para serviços *Web*.
- Reduz potenciais erros através da diminuição de código necessário, aumentando a simplicidade e eficiência da implementação dos serviços *Web* do lado da aplicação cliente.
- Facilita alterações ao nível de código. A descoberta dinâmica de descrições WSDL permite que as alterações possam ser transmitidas automaticamente às aplicações cliente através de WSDL de forma a que não sejam necessárias modificações na aplicação cliente sempre que o código do serviço sofra modificações.

Cada documento WSDL obedece a uma estrutura bem definida, composta pelos seguintes elementos [57]

- *definitions*: elemento raiz do documento. Define os atributos *name*, para atribuir um nome ao documento e vários *namespaces* utilizados ao longo do documento.
- *documentation*: utilizado para a inserção de comentários em texto ou XML, para serem interpretados por humanos.
- *types*: descrever todos os tipos de dados trocados entre o cliente e o servidor. WSDL não tem definida uma sintaxe específica mas utiliza por defeito a sintaxe da linguagem *W3C XML Schema*
- *message*: este elemento define mensagens que podem ser de pedido (*input*) ou de resposta (*output*), possui um atributo *name* que define o nome da mensagem e possui ainda um conjunto de parâmetros (*part*) de entrada e de resultados de mensagens.
- *operation*: este elemento permite associar pares de mensagens do tipo pedido com a respetiva mensagem do tipo resposta.
- *portType*: este elemento combina múltiplos elementos *message* para formar uma operação disponibilizada pelo serviço *Web* e mensagens envolvidas. De notar que este elemento frequentemente define várias operações.
- *binding*: permite definir qual o protocolo de transporte utilizado na troca de mensagens SOAP associadas a cada *portType*, sendo o protocolo HTTP o mais utilizado.
- *service*: define o nome e o endereço do serviço a ser invocado através de um conjunto de elementos *port*.

É importante referir que a especificação WSDL não vem introduzir nenhuma definição nova, contudo reconhece a necessidade de sistemas mais elaborados para a descrição de mensagens e suporta a especificação XML *Schema Specification*.

A geração WSDL é criada automaticamente pelo próprio *software* utilizado na criação do serviço. Este processo está representado na figura 4.9. Na figura, no passo 1, o criador do serviço *Web* cria uma descrição WSDL através da ferramenta de desenvolvimento que analisa a camada de interface SOAP do serviço *Web*. No passo seguinte, a aplicação cliente gera o código necessário para a comunicação com o serviço *Web* através da análise

da descrição WSDL. No último passo, a aplicação cliente e o serviço *Web* encontram-se em condições para poderem comunicar entre si [5].

Figura 4.9: WSDL como mecanismo para facilitar a comunicação entre serviços *Web* e outras aplicações [5]

4.4 UDDI

UDDI é uma especificação técnica para descrever, descobrir e integrar serviços *Web* através da criação de repositórios, onde os serviços *Web* podem ser pesquisados e registados por fornecedores e utilizadores [2].

Apesar de ser possível realizar o acesso direto a um serviço *Web* através da sua descrição WSDL, tal só se torna possível se se conhecer à partida onde está alojado o serviço *Web*, através do seu URL, para encontrar serviços *Web* novos ou para divulgar a oferta de um serviço *Web* recorre-se a um sistema de catálogo [58].

Um exemplo de um registo de registos é o *UDDI Business Regestry* (UBR), suportado por empresas como IBM, *Microsoft* e SAP, entre outras. Qualquer entidade pode publicar ou pesquisar serviços *Web*, gratuitamente.

A informação capturada pelo UDDI pode ser dividida em três categorias principais, figura 4.10 [54]:

- *Páginas Brancas*: contém informação geral sobre o criador que fornece o serviço *Web*,nome, morada, telefone, descrição da empresa, entre outros.

- *Páginas Amarelas*: contém a informação do serviço *Web* e do seu fornecedor registada em categorias.

- *Páginas Verdes*: contém a informação técnica sobre o serviço *Web*. Na maioria dos casos, inclui um ponteiro para uma especificação externa e um endereço para a invocação do serviço *Web*. Descreve o comportamento e funções disponibilizadas pelo *Web Service*, assim como a sua localização.

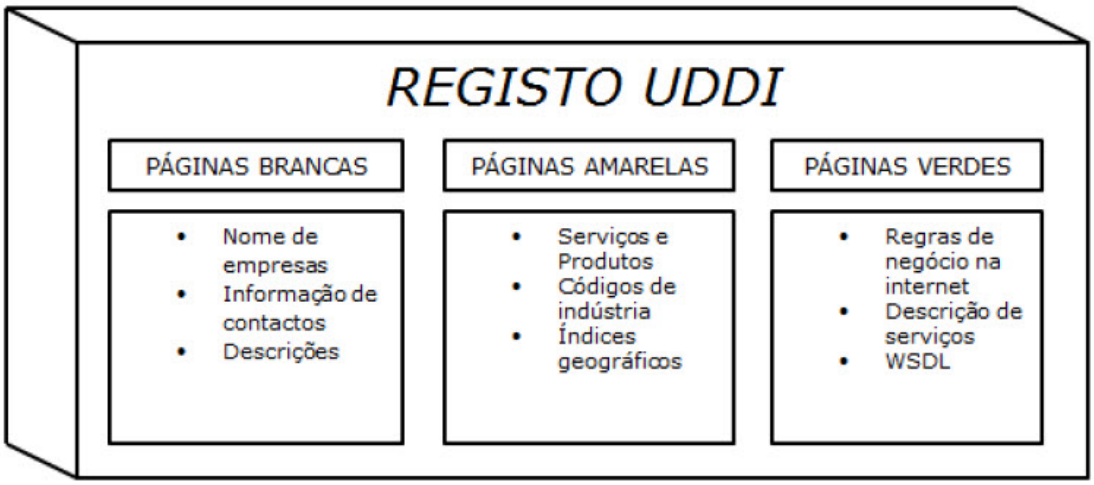


Figura 4.10: Representação de um registo UDDI [5]

A aplicação da especificação UDDI não se restringe apenas a aplicação em serviços *Web* baseados em SOAP. A especificação pode ser utilizada para descrever qualquer serviço desde uma página *Web* ou endereço de email até CORBA.

O UDDI é um componente essencial na descrição dos serviços *Web*, permitindo a criação, especificação, descoberta e invocação dos serviços.

Em ambiente empresarial pode ser uma mais valia o uso de um registo privado. O uso deste registo poderá possibilitar uma comunicação entre serviços distintos, minimizando o tráfego de documentação atualmente existente entre as empresas e corporações, muitas das vezes pertencentes ao mesmo grupo. Assim que os serviços são submetidos no registo privado UDDI, passa a existir a possibilidade de conectar dinamicamente projetos e aplicações dentro da mesma organização [5].

4.5 HTTP

O *Hipertext Transfer Protocol* (HTTP) é a base da transmissão de dados através da *World Wide Web*. Refere-se ao protocolo de comunicação responsável pelo transporte da grande maioria das aplicações *Web* e baseia-se em pedidos e respostas entre clientes e servidores. O cliente uma mensagem para o servidor onde se encontram os recursos e conteúdos, ou pode realizar outras funções de interesse. Após a conclusão do processo o servidor devolve uma resposta ao cliente. A resposta pode conter informações de conclusão de operações requisitas ou conter conteúdo solicitado no corpo da mensagem.

Considerando como exemplo o acesso a uma página Web simples, programada em *Hipertext Markup Language* (HTML) , o servidor HTTP (*Web Server*) aloja a informação da página, como texto, imagens, enquanto que o cliente pode ser um Web Browser. Quando o cliente faz um pedido, o servidor responde com o código HTML inserido numa mensagem HTTP, sem necessitar de ter muita informação acerca do cliente. O protocolo HTTP faz uso dos protocolos TCP/IP, sendo a porta TCP a número 80 por omissão.

No contexto deste trabalho, são principalmente relevantes os métodos *Post* para submeter os dados e o método *Get* responsável pela devolução dos dados durante a implementação dos serviços *Web* [56].

4.6 IP

O protocolo IP foi desenvolvido pelo departamento da defesa dos Estados Unidos no âmbito do projeto *Defence Advanced Research Projects Agency* (DARPA).

Foi desenvolvido para permitir a troca de blocos de dados entre computadores. Cada equipamento tem um endereço definido pelo próprio protocolo, de forma a permitir encaminhar os blocos de dados de equipamento em equipamento até ao equipamento de destino. Este protocolo permite também a fragmentação dos dados a enviar em mensagens mais pequenas, que possam ser transmitidas através de redes locais como *Ethernet*. Permite ainda a o envio de dados de forma independente da(s) rede(s) físicas que existem entre eles, criando para isso uma rede virtual baseada em endereços Internet.

O protocolo IP fornece à camada de transporte um serviço de transferência de informação não confirmado, sem estabelecimento de ligação levando a que a troca fiável de mensagens ente as camadas de transporte não seja garantida. O percurso das mensagens através dos vários sistemas intermédios não é constante, podendo por isso chegarem fora de ordem ao destino [56].

4.7 TCP

Os *Transmission Control Protocol* (TCP), assim como o protocolo IP, foi desenvolvido pela *DARPA* e em conjunto com o protocolo IP forma a base que serve de suporte à rede de computadores que forma a Internet.

O protocolo IP fornece um serviço não confirmado de entrega de mensagens levando a possibilidade da ocorrência mensagens que chegam ao destino fora de ordem ou não cheguem de todo.

O protocolo TCP fornece um serviço fiável de transferência de dados, apesar de utilizar os serviços prestados pelo protocolo IP. Está presente nos equipamentos de origem e destino que estabelecem entre si uma ligação virtual, onde a receção dos dados é sempre confirmada pelo recetor e permitem a retransmissão automática das mensagens perdidas. Este protocolo também permite que várias aplicações no mesmo equipamento comuniquem através da mesma ligação de rede IP. Para isso as várias ligações de transporte são identificadas pelo número da porta virtual utilizada. O protocolo TCP garante que não há perda de pacotes e que chegam ao destino na ordem correta.

Trata-se de um protocolo muito versátil e robusto, adequado para grandes redes globais pois garante a entrega dos pacotes de dados e implementa a camada de transporte para o suporte do protocolo HTTP [56].

Capítulo 5

Solução proposta

Neste capítulo são apresentadas as diferentes arquiteturas propostas para a execução do presente trabalho. É ilustrada a arquitetura global de ligação entre os componentes envolvidos no sistema e a arquitetura do sistema capaz de fazer a organização das ordens de fabrico onde se insere o *software* proposto e a estrutura do mesmo. Em seguida, são descritos todos os processos utilizados na implementação dos elementos da arquitetura, bem como aspetos relativos às tecnologias adotadas na construção da solução.

5.1 Arquitetura do sistema

O objetivo do presente trabalho passa por fazer uso da tecnologia dos serviços *Web* com vista a uma melhor integração dos sistemas de produção, desde os sistemas superiores de planeamento e escalonamento aos sistemas de nível inferior como os recursos produtivos.

Procura-se desenvolver uma estrutura onde o utilizador possa facilmente introduzir os dados e visualizar os seus resultados. Após a verificação dos resultados o cliente pode optar por dar seguimento às ordens introduzidas.

Pretende-se implementar uma arquitetura com dois componentes:

- **Interface gráfica** - a aplicação deverá ser capaz de exibir a informação respeitante à introdução das ordens de fabrico;

- **Serviço *Web*** - serviço que será responsável pela comunicação com uma base de dados para obter a lista de prioridades pela qual deverá organizar a lista de ordens introduzidas pelo cliente. Também será responsável por efetuar o cálculo de otimização através de um algoritmo para encontrar a melhor solução à redução dos trabalhos atrasados com pesos atribuídos.

A arquitetura proposta encontra-se representada na figura 5.1. Esta arquitetura pretende fazer uso da tecnologia dos serviços *Web* de forma a preparar uma possível implementação futura de uma arquitetura SOA.

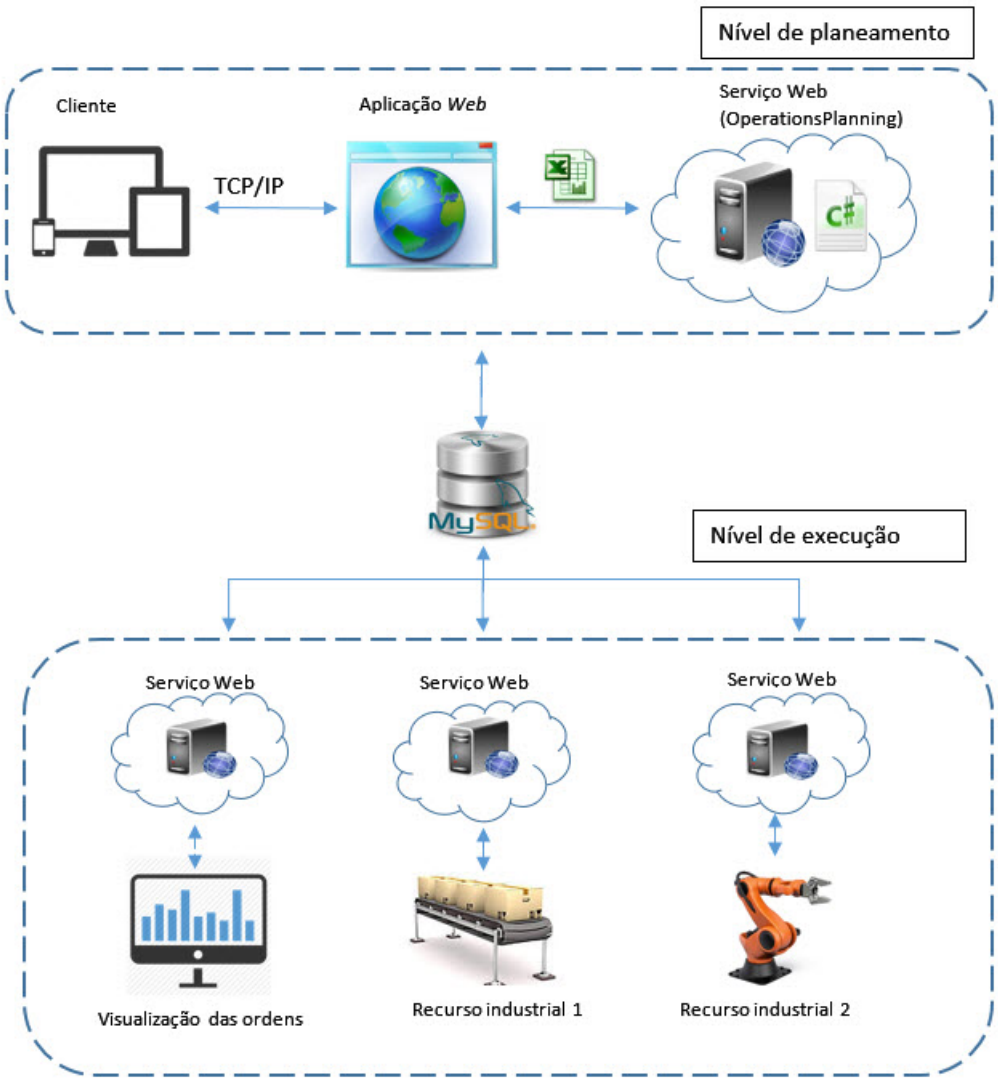


Figura 5.1: Arquitetura do sistema proposto

É formada por uma base de dados central onde são armazenadas todas as informações do nível superior responsável pelo planeamento assim como os estados dos diapositivos distribuídos do nível de execução ou nível inferior. O nível de planeamento será composto por uma aplicação *Web* que recebe o ficheiro das ordens e procede à invocação do serviço *Web* ProductionPlanning para a execução do sequenciamento das tarefas contidas no ficheiro. Após a conclusão da operação, todas as informações são armazenadas na base de dados e tornam-se disponíveis para os restantes níveis do sistema. Posteriormente, os recursos do nível de execução consultam as informações contidas na base de dados para ficarem a conhecer a sequencia pela qual as tarefas devem ser processadas.

Neste trabalho apenas será abordado o nível de planeamento. Na perspetiva de proporcionar e facilitar a integração com mais sistemas informáticos, pretende-se propor o serviço *Web* ProductionPlanning que será responsável pela leitura do ficheiro das ordens de fabrico e pelo sequenciamento dessas ordens segundo o método de prioridades da empresa e um método de otimização através de algoritmos genéticos.

5.2 Arquitetura do serviço *Web* proposto

A arquitetura proposta na figura figura 5.2 pretende organizar as ordens de fabrico segundo uma lista de prioridades definida por parte da empresa, com vista a automatizar o processo e reduzir a probabilidade de erro humano.

O sistema é composto por um serviço *Web* responsável pelo cruzamento das informações entre uma lista de prioridades e a lista das ordens de fabrico que são introduzidas na produção, contidas num ficheiro *Excel*.

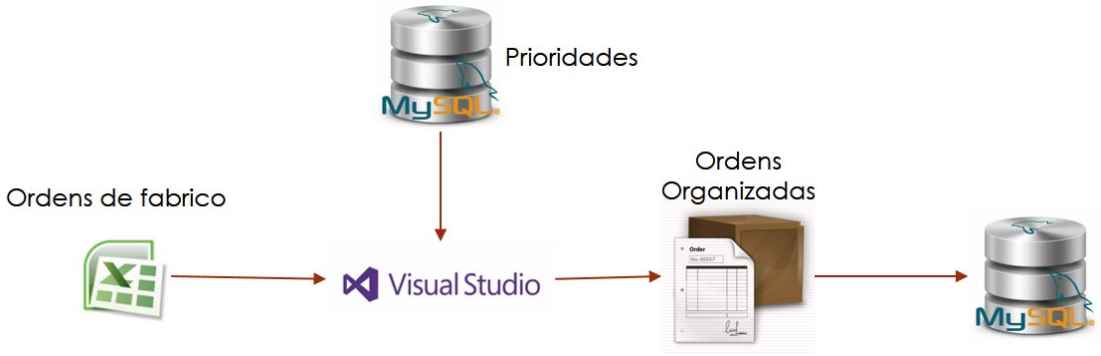


Figura 5.2: Arquitetura do sistema de organização segundo lista de prioridades

No ficheiro *Excel* estão descritas as ordens de fabrico, juntamente com outros dados adicionais como quantidades, nome do cliente, referência do produto, entre outras informações necessárias a cada ordem, como pode ser observado na figura 5.3.

QT.	REF.	CARIMBO	CLIENTE	S.	OBS.	N.º	Data
400	7715/A T/L 35	GNS/ECO	GNS	40			27-06-2014
200	7715 T/L 35	GNS/ECO	GNS	42			27-06-2014
60	7715/A T/L 35	DEM	GNS	41		1169 D	27-06-2014
60	7715 T/L 35	GNS/ECO	GNS	41		1167 D	27-06-2014
260	7715/A T/L 35	GNS/ECO	GNS	40		1193D,1197D	27-06-2014
6	7715 T/L 30	GNS/ECO	UA	42	- 5#,	1177 D	27-06-2014
12	7715 T/L 35	DEM	UA	42	- 5#,	1178 D	27-06-2014
30	7715 T/L 35	GNS	UA	41	- 5#,	1179 D	27-06-2014
30	7799 T/U 25	GNS	UA	41	- 5#,	1191D	27-06-2014
30	7800 T/U 25	UAN	UA	42	- 5#,	1192 D	27-06-2014
80	7716/A T/L 35	UAN	GNS	42	- 5#,	1183 D	27-06-2014
30	7802 T/U 35	UAN	GNS	42	- 5#,	1181 D	27-06-2014
50	7803 T/L 30	UAN	GNS	42	- 5#,	1180 D	27-06-2014
30	7804 T/L 35	UAN	GNS	42	- 5#,	1184 D	27-06-2014
30	7716 T/L 35	UAN	GNS	42	- 5#,	1186 D	27-06-2014
6	7716 T/L 35	UAN	GNS	42	- 6#,	1186 D	27-06-2014
12	7716 T/L 35	UAN	GNS	42	- 6#,	1188 D	27-06-2014
2	7717 T/U 30	UAN	GNS	42	- 6#,	1189 D	27-06-2014
6	7717 T/L 35	UAN	UA	42	- 6#,	1185 D	27-06-2014
6	7799 T/U 25	UAN	UA	42	- 6#,	1190 D	27-06-2014
2	7796 T/L 35	GNS	UA	41	- 2,10	1187 D	27-06-2014
2	7799/A T/L 30	GNS	UA	41	- 2,10	1203 D	27-06-2014
2	7799/A T/L 35	GNS	UA	41	- 2,10	1204 D	27-06-2014

Figura 5.3: Ficheiro Excel com as ordens de fabrico

A lista de prioridades fica ao encargo de uma base de dados onde serão listadas as prioridades a serem utilizadas pelo serviço *Web*. Posteriormente a lista pode ser modificada sem necessidade de qualquer alteração na programação do serviço.

Assim que o serviço receba as informações das ordens de fabrico e da lista de prioridades, este faz o cruzamento da informação, devolvendo a lista organizada. Obtida a

lista organizada, as ordens são armazenadas numa base de dados, permitindo a sua consulta e possível monitorização para outras operações dentro da empresa. Posteriormente pode-se fazer uso do escalonamento a partir do algoritmo genético por forma a encontrar a melhor combinação de ordens para a redução dos trabalhos em atraso.

Este é o conceito de sistema explorado no presente trabalho. Para o consumo do serviço *Web*, foi desenvolvida uma interface gráfica através da linguagem HTML5 que pode ser executada a partir de qualquer browser de acesso a páginas de Internet.

5.3 Construção do sistema de prioridades

A solução é composta por um serviço e um cliente em forma de aplicação web, para demonstrar o consumo do serviço. A decisão de uma aplicação web passou pela possibilidade de esta poder ser acedida a partir de qualquer dispositivo com acesso a Internet. No entanto a abordagem a uma arquitetura SOA baseada em serviços *Web* possibilita a incorporação dos métodos do serviço não só em ambiente *Web*, mas em qualquer tipo de aplicações capazes de consumir os serviços, independente do sistema operativo, plataforma de desenvolvimento ou linguagem de programação.

Como linguagem de programação optou-se pela linguagem C# através do *Visual Studio 2012*. *C#* é uma linguagem de tipos protegidos, orientada a objeto e que permite aos programadores construírem uma variedade de aplicações seguras e robustas, compatíveis com o *.NET Framework*. É possível usar esta linguagem para criar aplicações de cliente em ambiente *Windows*, serviços *Web* XML, componentes distribuídos, aplicações do tipo cliente-servidor, aplicações de interações com bases de dados, entre outras.

Algumas das características mais importantes da linguagem *C#* podem ser [59]:

- Simplicidade, os programadores muitas vezes fazem várias referências de que se trata de uma linguagem tão poderosa como a linguagem *C++* e tão simples como a linguagem *Visual Basic*
- Completamente orientada a objetos onde qualquer variável tem, obrigatoriamente de fazer parte de uma classe;
- Fortemente tipada o que ajuda a evitar erros na manipulação imprópria de tipos e atribuições incorretas;
- Flexibilidade pois possibilita a utilização de ponteiros pelos programadores

Foi então desenvolvido o serviço *Web ProductionOperations* formado por três métodos que podem ser vistos na figura 5.4. Os métodos expostos pelo serviço são respetivamente: **ProductionOptimizationAlgorithm**, **ReadExcelFile** e **sortByPriorities**. Cada um dos métodos apresenta uma breve descrição do seu objetivo bem como os parâmetros que tem de receber para poder ser invocado.

O método **ReadExcelFile** foi desenvolvido para fazer a conversão das ordens introduzidas através de um ficheiro *Excel*, por parte da empresa, para objetos na linguagem de programação por forma a poderem ser trabalhados e manipulados. Este, faz a leitura de cada uma das linhas da tabela como sendo uma ordem cujos campos do objeto são automaticamente preenchidos com os valores do ficheiro introduzido.

Após a utilização do método anterior, encontram-se reunidas as condições para a utilização dos dois métodos restantes. No caso do método **sortByPriorities**, este faz a

organização da tabela das ordens segundo uma lista de prioridades definida, devolvendo a lista organizada assim que termina a sua execução. O método **ProductionOptimizationAlgorithm** também executa uma organização das ordens de produção contudo desta vez através da utilização de um AG que devolve a solução ótima para a otimização em questão.

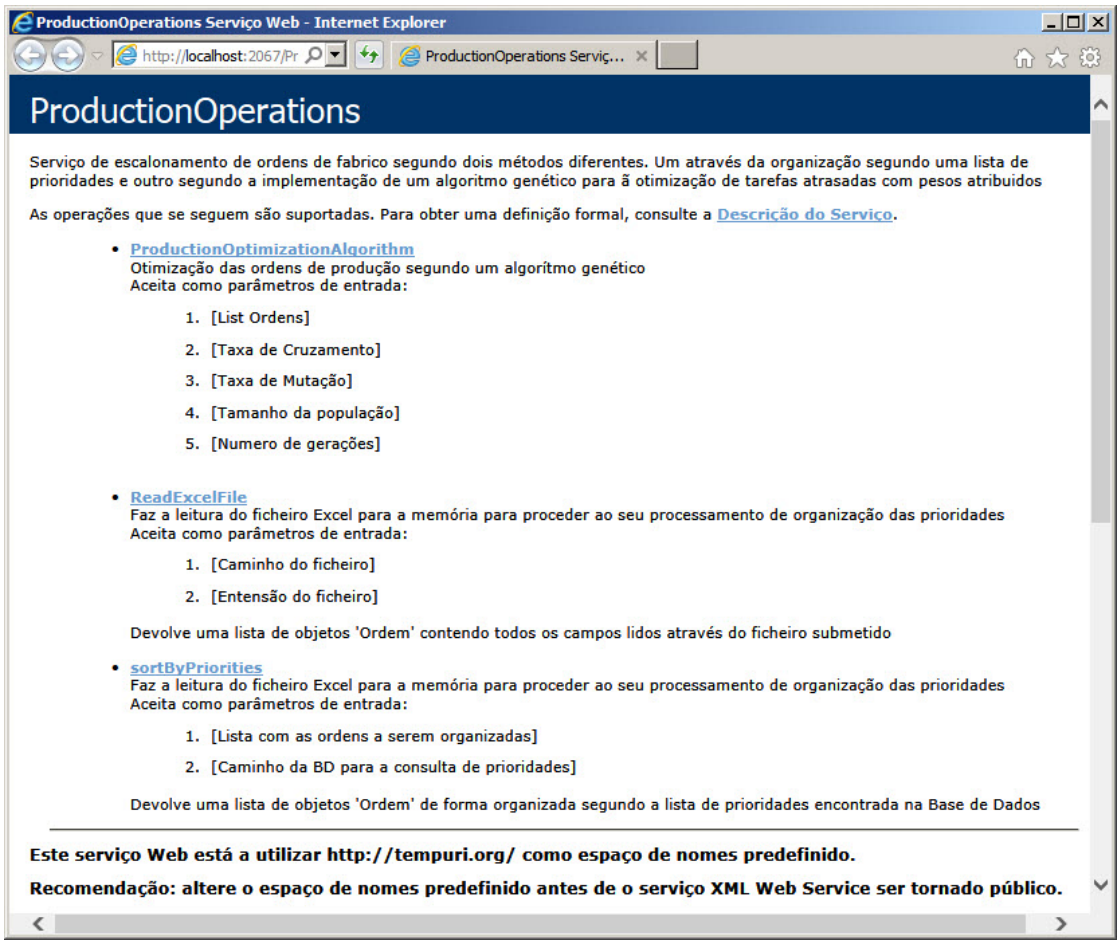


Figura 5.4: Página *Web* com os métodos expostos pelo serviço *Web*

A descrição WSDL do serviço e dos métodos desenvolvidos pode ser vista em parte, na figura 5.5. É a partir deste documento XML, que a aplicação cliente tem acesso à interface do serviço *Web*, onde se encontram descritas todas as suas funcionalidades, troca de mensagens, parâmetros a utilizar e os documentos de resposta.

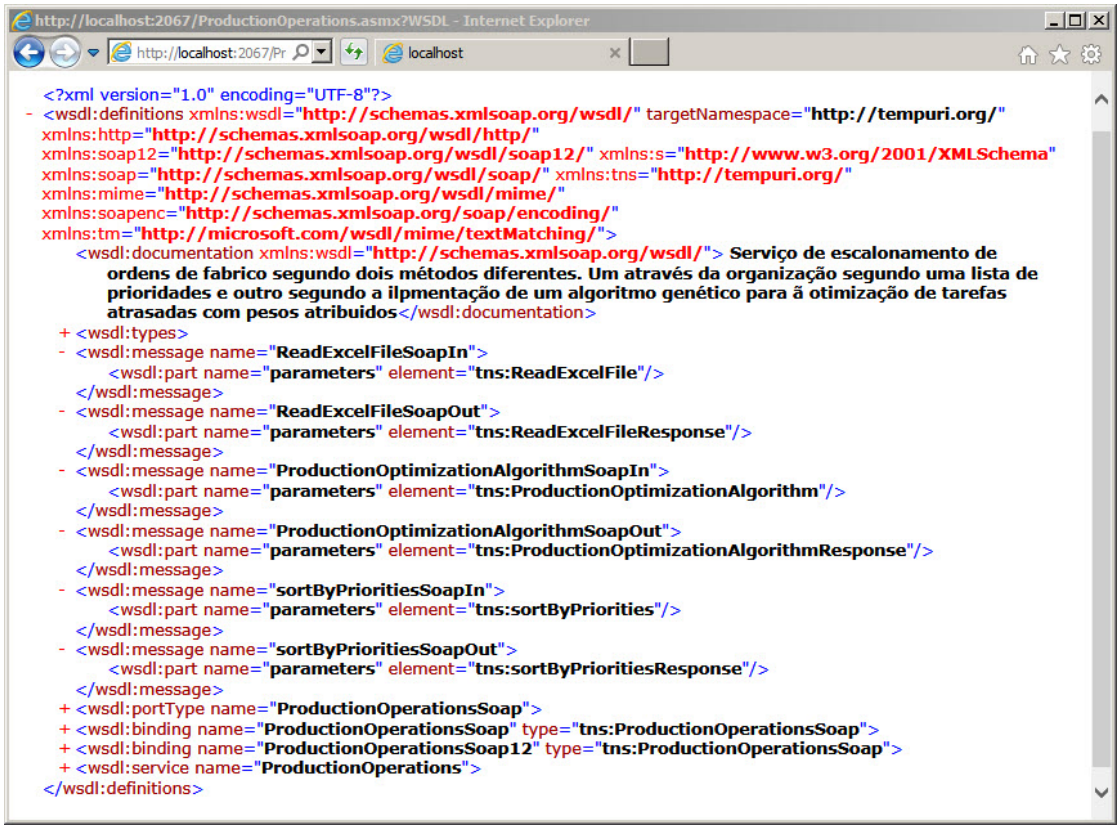


Figura 5.5: Descrição WSDL do serviço desenvolvido

No documento WSDL identificam-se com alguma facilidade os principais elementos do documento: *definitions*, *documentation* onde se encontra uma pequena descrição para os utilizadores humanos, *types*, vários elementos *message* de *input* e *output* que se encontram emparelhados nos elementos de *binding*, *portType* e o *service*.

Respeitante ao consumo do serviço, este faz-se através da troca de mensagens XML, de acordo com a especificação SOAP. Contudo, apesar do protocolo SOAP não definir um protocolo de transporte, o protocolo HTTP tem sido o mais implementado.

De seguida analisa-se uma mensagem SOAP, figura 5.6, trocada entre o cliente e o servidor durante a invocação do método *ReadExcelFile*, utilizando o protocolo HTTP como protocolo de transporte da mensagem.

Da análise do código, verifica-se que as primeiras cinco linhas correspondem à informação referente ao protocolo HTTP onde:

- Linha 1: contém a informação do método (POST), do URI, do protocolo (HTTP) e a sua versão;
- Linha 2: indica o URL do lado do servidor onde o serviço se encontra alojado;
- Linha 3: define o conteúdo da mensagem e do tipo de texto;
- Linha 4: fornece informação sobre o tamanho da mensagem;
- Linha 5: indica que se trata de uma mensagem SOAP onde está definido o URI a ser invocado.

```
1 POST /ProductionOperations.asmx HTTP/1.1
2 Host: localhost
3 Content-Type: text/xml; charset=utf-8
4 Content-Length: length
5 SOAPAction: "http://tempuri.org/ReadExcelFile"
6 <?xml version="1.0" encoding="utf-8">
7 <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance"
8 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
10  <soap:Body>
11    <ReadExcelFile xmlns="http://tempuri.org/">
12      <FilePath>string</FilePath>
13      <Extension>string</Extension>
14    </ReadExcelFile>
15  </soap:Body>
16 </soap:Envelope>
```

Figura 5.6: Mensagem SOAP para invocar o método *ReadExcelFile*

As restantes linhas correspondem à informação que é transferida através da mensagem SOAP:

- Linha 6: indica o início do documento XML onde se pode encontrar a versão e o tipo de codificação utilizado pelo protocolo;
- Linha 7: abertura do elemento *Envelope*;
- Linha 8 e 9: fazem referência aos URI's dos *namespaces*
- Linha 10: inicio do elemento *Body*;
- Linha 11: indica o nome do método a ser invocado no serviço;
- Linha 12, 13 : fazem referencia aos parâmetros que devem ser introduzidos para a invocação do método;
- Linha 14: indica a conclusão da invocação do método;
- Linha 15: fecho do elemento *Body*;
- Linha 16: fecho do elemento *Envelope*

Neste caso não se encontra presente o elemento opcional <*soap:Header*>. Os componentes mais importantes da mensagem SOAP encontram-se nas linhas 12, 13 e 14 que correspondem ao parâmetros de entrada necessários fornecer ao método para a sua execução e devolução dos resultados. Da análise do código pode-se verificar o que método espera dois parâmetros de entrada do tipo *string*, uma com o caminho do ficheiro e outro com a extensão do ficheiro submetido.

Após a execução do método, a mensagem SOAP de resposta pode ser consultada na figura 5.7. Esta será a mensagem enviada para o cliente que deverá ser descodificada na aplicação cliente de forma a mostrar os resultados de uma forma de leitura mais fácil para o utilizador, possivelmente através da utilização de uma interface gráfica.

```
1  HTTP/1.1 200 OK
2  Content-Type: text/xml; charset=utf-8
3  Content-Length: length
4  <?xml version="1.0" encoding="utf-8">
5  <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance"
6  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
       schemas.xmlsoap.org/soap/envelope/">
7    <soap:Body>
8      <ReadExcelFileResponse xmlns="http://tempuri.org/">
9        <ReadExcelFileResult>
10         <Orders>
11           <Peso>double</Peso>
12           <qt>string</qt>
13           <Ref>string</Ref>
14           <Cliente>string</Cliente>
15           <DataEntrega>dateTime</DataEntrega>
16           <Carimbo>string</Carimbo>
17           <time2Process>double</time2Process>
18         </Orders>
19         <Orders>
20           <Peso>double</Peso>
21           <qt>string</qt>
22           <Ref>string</Ref>
23           <Cliente>string</Cliente>
24           <DataEntrega>dateTime</DataEntrega>
25           <Carimbo>string</Carimbo>
26           <time2Process>double</time2Process>
27         </Orders>
28       </ReadExcelFileResult>
29     </ReadExcelFileResponse>
30   </soap:Body>
31 </soap:Envelope>
```

Figura 5.7: Mensagem SOAP de resposta ao método *ReadExcelFile*

As primeiras 3 linhas correspondem a informações sobre o protocolo. Se durante o envio do pedido SOAP fez-se uso do método *POST*, a reposta ao pedido é devolvida através do método *GET*. Analisando a mensagem verifica-se que o elemento *Body* devolve os resultados do método. Os resultados são expressos a partir da linha 9 até à linha 28, através da etiqueta *<ReadExcelFileResults>*. Entre estas linhas verifica-se que o resultado é devolvido segundo a estrutura *<Orders>* formada pelos campos *<Pesos>*, *<qt>*,

<Ref>, *<Cliente>*, *<DataEntrega>*, *<Carimbo>*, *<time2Process>* que correspondem respetivamente aos pesos, quantidades, referências, nomes dos clientes, datas de entrega e tempos de processamento.

O exemplo da mensagem SOAP na figura 5.7 exemplifica o código XML sobre o qual os resultados são devolvidos à aplicação cliente. Na imagem 5.8 pode-se visualizar uma resposta do método após a leitura do ficheiro das ordens de produção.

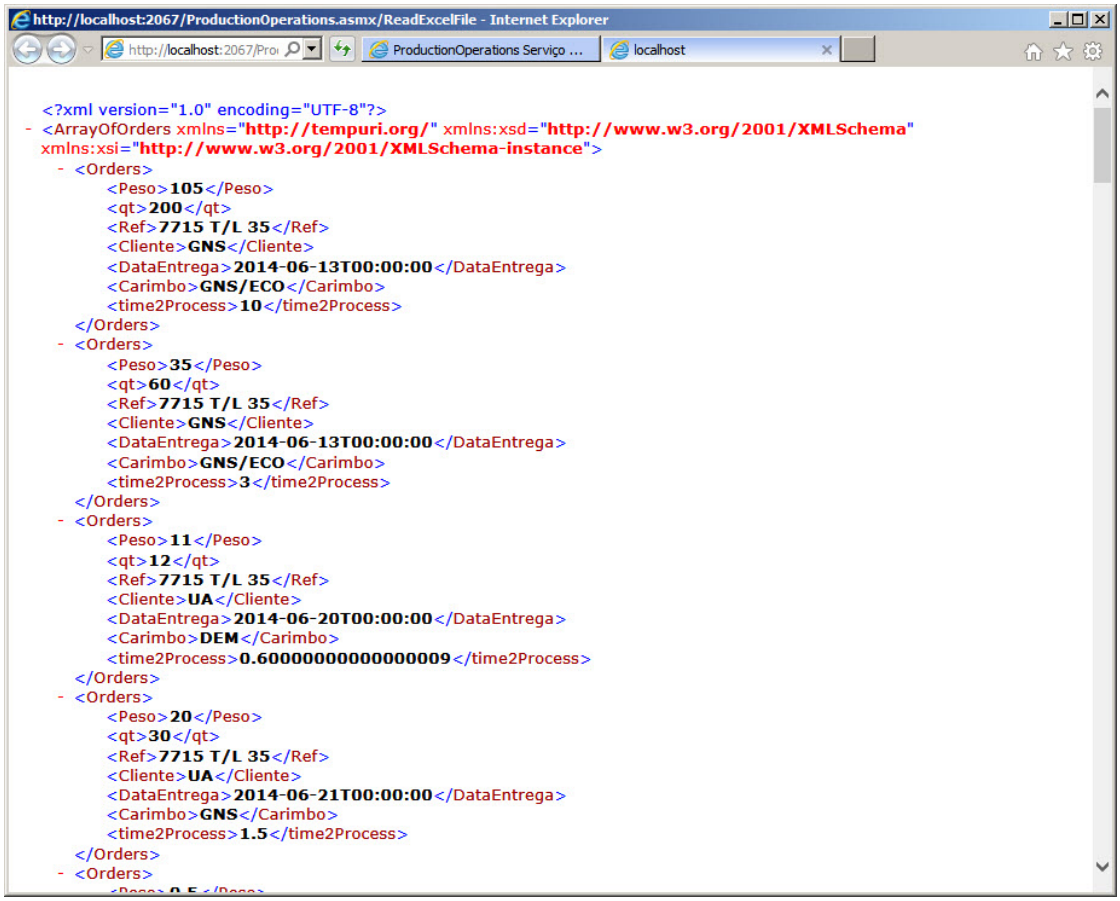


Figura 5.8: Resposta em XML após a invocação do método *ReadExcelFile*

Como já foi referenciado, o serviço *Web* desenvolvido pode ser consumido em diferentes tipos de aplicações, independentemente do sistema operativo, plataforma de desenvolvimento ou linguagem de programação. Nesse sentido, foi desenvolvida uma aplicação *Web* para demonstrar o consumo do serviço desenvolvido. Na figura 5.11 pode ver-se a interface gráfica da aplicação *Web* desenvolvida.

Para o desenvolvimento da aplicação *Web* foi utilizada a linguagem *Active Server Page* (ASP) com a linguagem do lado do servidor em *C#* e *Framwork .Net*.

Dentro do projeto da aplicação em *Visual Studio* adiciona-se o serviço *Web* através do menu *Add Service Reference* (figura 5.9), fornecendo o URL no caso de ser conhecido. Na janela podem ser vistos os métodos expostos pelo serviço e o nome a ser atribuído para a sua utilização no projeto, que no caso da figura é *ServiceReference1*.

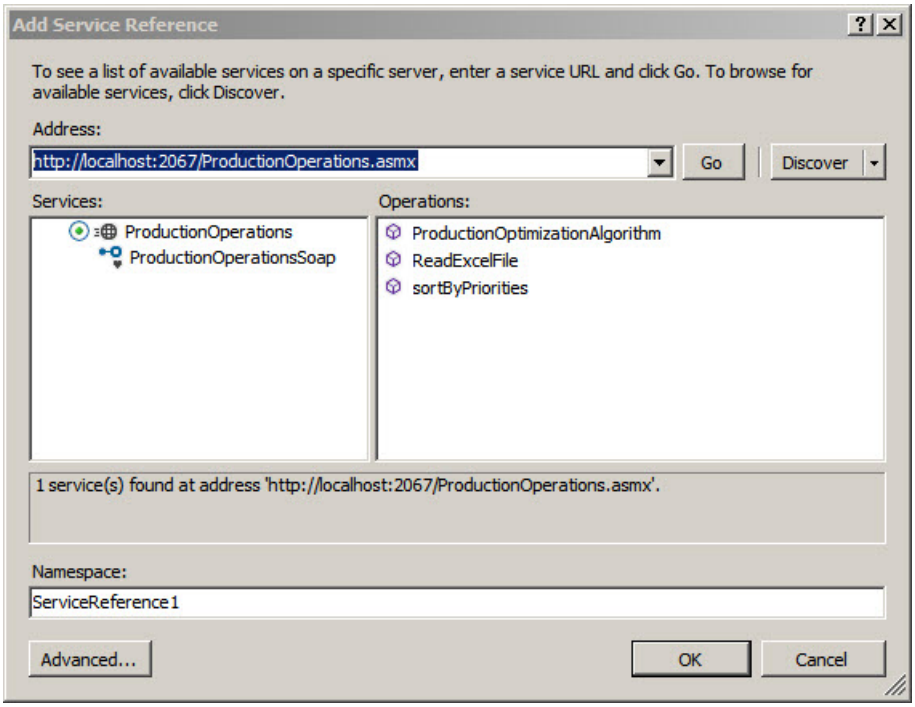


Figura 5.9: Adição do serviço *Web* ao projeto

Logo que adicionado, o serviço passa a ficar disponível no ambiente de desenvolvimento como se de um objeto de tratasse à espera de ser invocado. Na figura 5.10 pode ser visto um exemplo da facilidade com que se invoca o serviço desenvolvido que processa o ficheiro das ordens de produção introduzido.

```
1  protected void bt_UploadFile_Click( object sender, EventArgs e )
2  {
3  // Obtém os dados do ficheiro submetido
4  string Filename =
       Path.GetFileName(Uploader_ExcelFile.PostedFile.FileName);
5  string FileExtension =
       Path.GetExtension(Uploader_ExcelFile.PostedFile.FileName);
6  string FilePath = Server.MapPath(FolderPath + Filename);
7  // Construtor do objeto SOAP
8  ServiceReference1.ProductionOperationsSoapClient ws = new
       ServiceReference1.ProductionOperationsSoapClient();
9  // Dados da tabela através da invocação do serviço
10 GridView1.DataSource = ws.ReadExcelFile(FilePath, FileExtension);
11 // Constrói a tabela com os dados recebidos
12 GridView1.DataBind();
13 }
```

Figura 5.10: Exemplo do código que invoca o serviço ReadExcelFile

O código mostra a forma simples da utilização dos serviços *Web*. Embora a sintaxe

do código possa ser diferente para outras linguagens de programação, o encapsulamento do código permite um elevado grau de abstração por parte do programador pois só precisa de fornecer os parâmetros requeridos pelo serviço e descodificar as respostas enviadas. O exemplo mostra a utilização do serviço **ReadExcelFile** onde são fornecidos os parâmetros do caminho e da extensão, obtidos através de um objeto *UploaderExcelFile*, utilizado de seguida na construção da aplicação *Web*.

Na aplicação *Web* a página é constituída por 3 campos, a barra de navegação responsável pela transição entre diferentes páginas *Web*, um campo para a introdução do ficheiros que contém as ordens de produção e um campo que mostra algumas informações genéricas acerca das ordens submetidas.

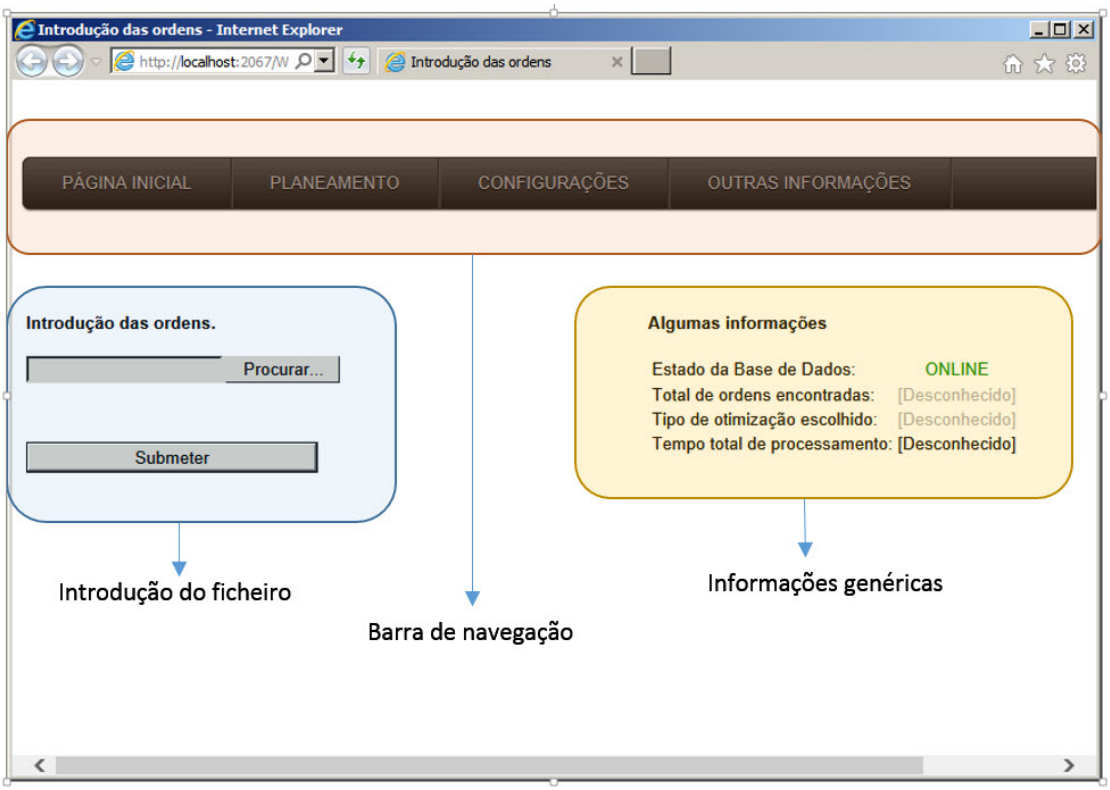
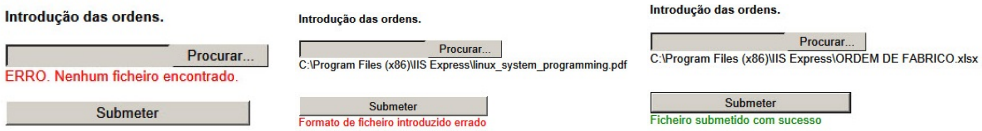


Figura 5.11: Página inicial da aplicação *Web*

O campo referente à introdução do ficheiro pode adotar formas diferentes conforme a sua utilização. Para o efeito foram previstos três casos possíveis:

- Caso 1: o utilizador pressiona o botão submeter sem a introdução de um ficheiro;
- Caso 2: o utilizador submete um ficheiro com o formato errado;
- Caso 3: o utilizador submete o formato de ficheiro correto para análise.

Na figura 5.12 podem ser vistos as respostas da aplicação *Web* para os três casos discutidos.



(a) Caso 1 (b) Caso 2 (c) Caso 3

Figura 5.12: Casos possíveis quando se submete um ficheiro

Assim que o utilizador submete o formato de ficheiro correto, a aplicação *Web* invoca o serviço *Web* para a organização das ordens de produção e apresenta-as ao utilizador (figura 5.13). A página passa a incluir uma tabela com as ordens lidas a partir do ficheiro, organizadas conforme a lista de prioridades, acompanhada de uma representação gráfica do horizonte temporal em termos de tempos de processamento.

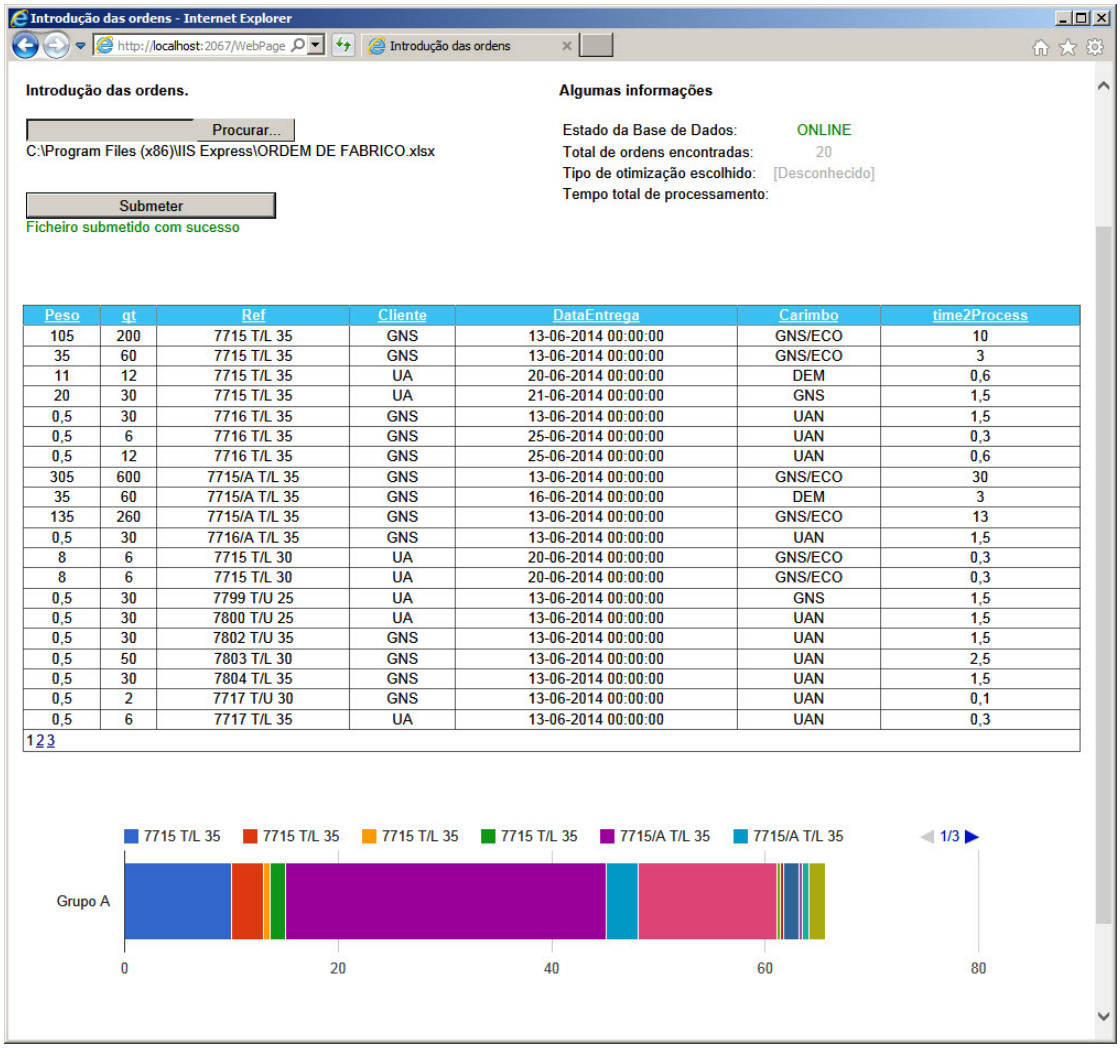


Figura 5.13: Interface da aplicação *Web* desenvolvida

5.4 Aplicação do AG ao sequenciamento da produção

O sequenciamento da produção está preocupado em determinar a ordem pela qual as tarefas devem ser processadas. No caso de processadores com carga moderada, onde as tarefas requerem o mesmo tempo de processamento, o escalonamento não apresenta dificuldades significativas. No entanto, para processadores que possuem cargas elevadas, a ordem pela qual as tarefas são processadas é muito importante em termos de custos associados às tarefas em espera de serem processadas.

No caso da empresa em questão, as tarefas referentes aos produtos do grupo A, são todas processadas através de um único processador, pelo que a definição da ordem dessas tarefas revela uma elevada importância em ser otimizada. Com o objetivo de otimizar o escalonamento foi implementado um AG que pretende minimizar o total de tarefas atrasadas com pesos atribuídos.

No problema da minimização dos trabalhos atrasados com pesos atribuídos, cada tarefa tem associada um tempo de processamento p_j, uma data de entrega d_j e um peso w_j que faz a diferenciação entre a importâncias das ordens num conjunto de ordens. Todas as tarefas estão disponíveis para processamento no instante $t = 0$, sendo que uma vez iniciadas não podem ser paradas até estarem terminadas. Para uma dada sequência de tarefas, o valor do atraso ocorre quanto o tempo de conclusão da tarefa j é maior que a sua data de entrega. O objetivo consiste em encontrar a sequência que minimiza o número de tarefas em atraso, $\sum w_j T_j$.

Dada a natureza combinatória do problema de sequenciamento recorreu-se à aplicação de um algoritmo genético. Nos próximos pontos serão explicados os aspetos e particularidades do algoritmo desenvolvido.

O algoritmo genético inicia-se através da criação de um grupo de indivíduos ou cromossomas formando um população, onde cada individuo representa uma possível solução para o problema. A população evolui através do operador cruzamento e operador mutação para procurar novas boas soluções.

5.4.1 Representação dos genes

Foi adotada a representação de valores reais, em que cada gene é formado por uma estrutura onde estão armazenados os valores de p_j, d_j e w_j como está representado na figura 5.14.

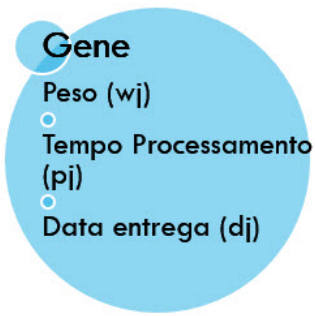


Figura 5.14: Representação dos elementos constituintes do gene

Assim consegue-se garantir que cada gene contém a informação referente a uma única

ordem de fabrico. As informações contidas no gene serão sempre mantidas durante a execução do algoritmo. Na figura 5.15 pode-se ver a estrutura desenvolvida informaticamente que é responsável para construção dos genes.

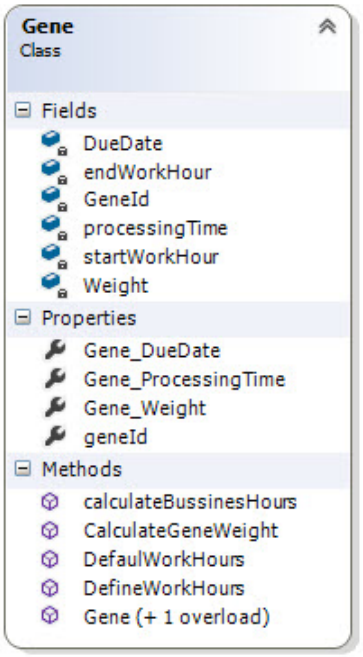


Figura 5.15: Estrutura informática desenvolvida para o gene

Verifica-se que o gene é composto pelas propriedades acima referidas que vão armazenar as informações retiradas das ordens de produção. Além das propriedades a estrutura possui ainda os campos de *startWorkHour* e *endWorkHour* que se referem respetivamente à hora de inicio e de termino do horário de trabalho da organização para a determinação do número de horas restantes até a data de entrega da ordem, calculadas através do método *calculateBuissinesHours*. Por definição é assumido um horário de trabalho de 8 horas entre as 09:00 horas e as 18:00 horas porém, caso seja necessário os valores podem ser modificados através da invocação o método *DefineWorkHours* que aceita como parâmetros de entrada a hora de inicio e fim do horário de trabalho.

Durante a construção do gene, outro paramento importante de salientar é a atribuição do peso da ordem a ser representada. Os pesos w_j são atribuídos de forma automática durante a construção do gene segundo a equação 5.1.

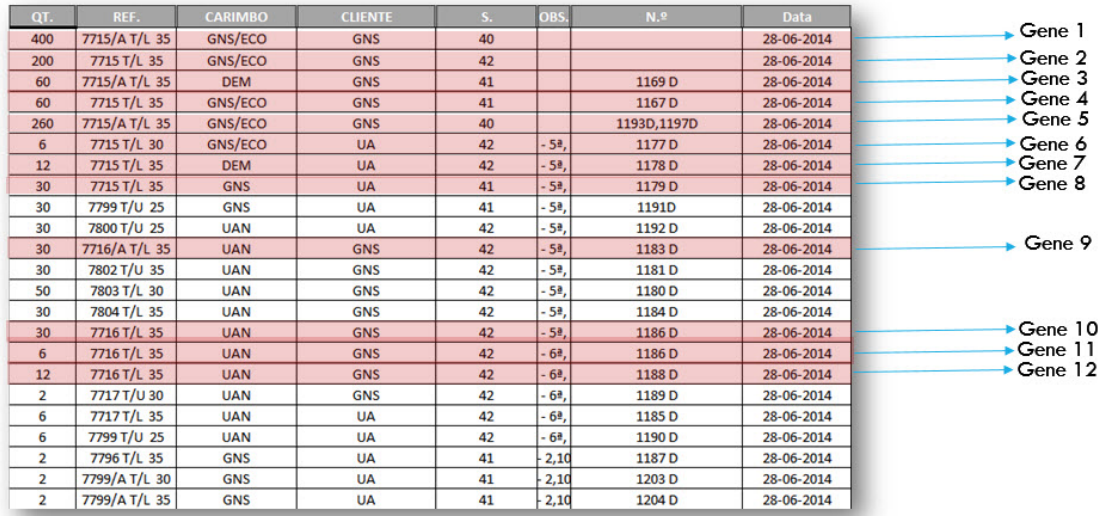
$$w_j = \alpha \cdot Quantidade_j + \beta \cdot Prioridade_j \tag{5.1}$$

Nesta equação são definidos dois parâmetros α e β que correspondem, respetivamente, à importância dada ao fator de quantidade e de prioridade. No caso da quantidade, esta é definida automaticamente pelo valor obtido da lista de ordens de produção. O valor do fator prioridades é definido numa escala de valores entre 12 e 2 sendo que o primeiro item da lista de prioridades recebe o valor de 12, o segundo item recebe um valor de 10, o terceiro item um valor de 8, o quarto item um valor de 6, o quinto um valor de 4 e o último item um valor de 2. Por definição foram atribuídos os pesos relativos de 25% para

o fator quantidade e 75% para o fator de prioridade, contudo e de forma semelhante à definição das horas de inicio e fim do horário de trabalho, os valores dos parâmetros α e β podem ser modificados através da invocação do método *CallculateGeneWeight*.

5.4.2 Construção da população do algoritmo

A população a ser utilizada pelo algoritmo é construída através da leitura dos elementos que fazem parte da lista de ordens de produção submetida pelo utilizador. Em primeiro lugar o algoritmo procura na lista as ordens que pertencem ao grupo de análise A, como se pode ver na figura 5.16. De seguida, cada linha encontrada corresponde a um gene cujos campos necessários para a formação da sua estrutura são obtidos da leitura dos parâmetros da ordem correspondente.



QT.	REF.	CARIMBO	CLIENTE	S.	OBS.	N.º	Data	
400	7715/A T/L 35	GNS/ECO	GNS	40			28-06-2014	Gene 1
200	7715 T/L 35	GNS/ECO	GNS	42			28-06-2014	Gene 2
60	7715/A T/L 35	DEM	GNS	41		1169 D	28-06-2014	Gene 3
60	7715 T/L 35	GNS/ECO	GNS	41		1167 D	28-06-2014	Gene 4
260	7715/A T/L 35	GNS/ECO	GNS	40		1193D,1197D	28-06-2014	Gene 5
6	7715 T/L 30	GNS/ECO	UA	42	- 5ª,	1177 D	28-06-2014	Gene 6
12	7715 T/L 35	DEM	UA	42	- 5ª,	1178 D	28-06-2014	Gene 7
30	7715 T/L 35	GNS	UA	41	- 5ª,	1179 D	28-06-2014	Gene 8
30	7799 T/U 25	GNS	UA	41	- 5ª,	1191D	28-06-2014	
30	7800 T/U 25	UAN	UA	42	- 5ª,	1192 D	28-06-2014	
30	7716/A T/L 35	UAN	GNS	42	- 5ª,	1183 D	28-06-2014	Gene 9
30	7802 T/U 35	UAN	GNS	42	- 5ª,	1181 D	28-06-2014	
50	7803 T/L 30	UAN	GNS	42	- 5ª,	1180 D	28-06-2014	
30	7804 T/L 35	UAN	GNS	42	- 5ª,	1184 D	28-06-2014	
30	7716 T/L 35	UAN	GNS	42	- 5ª,	1186 D	28-06-2014	Gene 10
6	7716 T/L 35	UAN	GNS	42	- 6ª,	1186 D	28-06-2014	Gene 11
12	7716 T/L 35	UAN	GNS	42	- 6ª,	1188 D	28-06-2014	Gene 12
2	7717 T/U 30	UAN	GNS	42	- 6ª,	1189 D	28-06-2014	
6	7717 T/L 35	UAN	UA	42	- 6ª,	1185 D	28-06-2014	
6	7799 T/U 25	UAN	UA	42	- 6ª,	1190 D	28-06-2014	
2	7796 T/L 35	GNS	UA	41	- 2,10	1187 D	28-06-2014	
2	7799/A T/L 30	GNS	UA	41	- 2,10	1203 D	28-06-2014	
2	7799/A T/L 35	GNS	UA	41	- 2,10	1204 D	28-06-2014	

Figura 5.16: Aquisição dos genes por parte do AG

Após a obtenção dos genes o algoritmo procede à construção do indivíduo (figura 5.17). Cada indivíduo corresponde a um solução possível onde a sequencia dos genes corresponde à sequencia das ordens de produção.

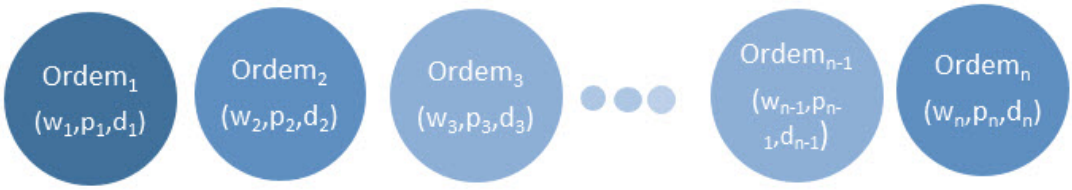


Figura 5.17: Representação de um indivíduo

Na figura 5.18 está representada a estrutura informática desenvolvida para a implementação e manipulação de um indivíduo.

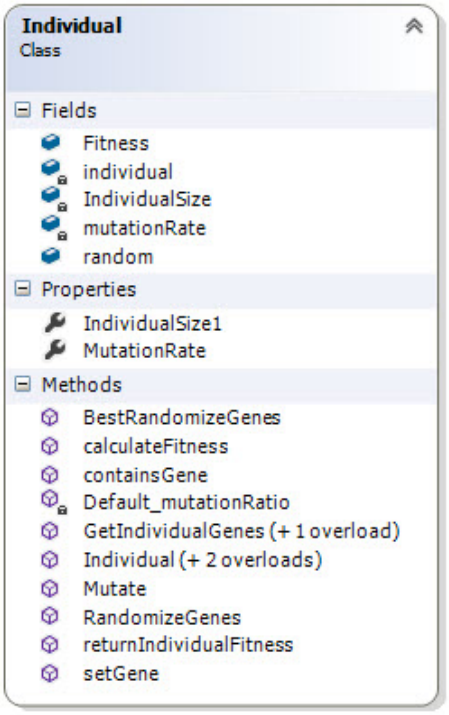


Figura 5.18: Classe do objeto indivíduo

Pode-se verificar que é nesta classe que é implementado o cálculo do valor de aptidão do indivíduo e que vai permitir ao algoritmo escolher os melhores indivíduos para as gerações seguintes. O cálculo da aptidão é executado através da invocação do método *calculateFitness* e foi programado de forma a ser executado de forma automática assim que o indivíduo é formado ou sofre algum tipo de operador. De destacar também que é nesta classe que é definida a taxa de mutação a ser imposta pelo operador de mutação através do método *Mutate*. Por definição foi implementada uma taxa de mutação de $1,5\%$ que, no entanto pode ser alterada.

O passo seguinte no algoritmo consiste na manipulação dos indivíduos para a criação de populações a partir das quais serão encontradas as melhores soluções para o problema. A população e as subsequentes gerações são compostas por um determinado número de indivíduos. Na figura 5.19 está representada a estrutura definida para a representação e manipulação da população de indivíduos do algoritmo. Para o presente trabalho utilizou-se uma população composta por 100 indivíduos. A população inicial é gerada de forma aleatória invocando o método *InitialPop*. Esta operação é executada o mesmo número de vezes que o tamanho definido para a população, sendo que de cada vez que é executado altera todas as posições de forma completamente aleatória dentro do indivíduo para formar um novo elemento da população.

Para avançar a população para a geração seguinte é invocado o método *CreateNextGeneration*. Os indivíduos das gerações seguintes provêm de duas fontes: reprodução onde indivíduos da população corrente são copiados para a nova geração e através da aplicação do operador cruzamento, ou seja, indivíduos que resultam da recombinação entre dois progenitores da população corrente. O operador cruzamento é representado pelo *Crossover* cuja a taxa de cruzamento usada no presente trabalho foi de 50% significando

que os restantes 50% dos melhores indivíduos da população são mantidos para a geração seguinte.

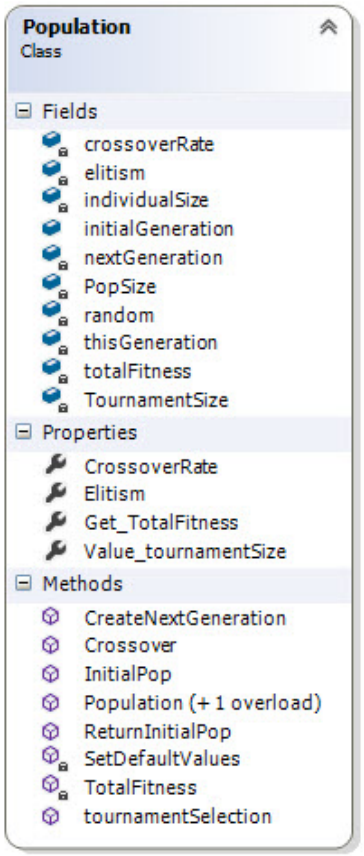


Figura 5.19: Classe do objeto população

Para a seleção dos indivíduos utilizados pelo operador cruzamento foi utilizado o método do torneio que é invocada pelo método *tournamentSelection*. Esta técnica simula a competição entre indivíduos aleatórios cuja a lista resultante é utilizada para hierarquizar os indivíduos servindo para determinar quais serão mantidos na geração seguinte e quais os indivíduos que serão eliminados. Dentro desse subgrupo os indivíduos competem entre si sendo o vencedor aquele com maior valor de aptidão e que apresenta maior possibilidade de obter uma geração de soluções ótimas. Os indivíduos selecionados serão utilizados pelo operador de cruzamento, de acordo com um valor de taxa de cruzamento, de ponto único para formar a descendência na nova geração.

Os indivíduos que não fazem parte do grupo dos melhores são sujeitos ao operador de mutação O operador mutação é responsável por manter a diversidade do material genético na população. Por se tratar de um problema de combinação de genes, adotou-se uma mutação por permuta de posição entre os genes. Desta forma é evitado o caso onde podem aparecer duplicações de elementos dentro do mesmo individuo. São geradas duas posições aleatórias que correspondem aos genes que sofrem a permuta.

Após a criação da nova geração de indivíduos esta substitui a geração anterior o número de vezes necessário ate que seja atingido o critério de paragem do algoritmo.

5.4.3 Critério de paragem

O AG move-se de uma geração para a geração seguinte através da seleção de progenitores para a criação de descendentes até que o critério de paragem seja verificado. Para se evitarem tempos de processamento demasiado longos, implementou-se um limite de 1000 gerações após as quais o algoritmo termina e devolve o indivíduo encontrado com o melhor valor de aptidão.

5.4.4 Resultados

O código da figura 5.20, demonstra um exemplo de como pode ser invocado o método responsável pela execução do algoritmo genético.

Uma vez que o algoritmo foi desenvolvido como um método pertencente ao serviço *Web* desenvolvido, este pode ser consumido facilmente por uma aplicação cliente. Para isso, apenas será necessário construir o objeto de comunicação SOAP (linha 2) e posteriormente invocar o método do serviço que se pretende com os parâmetros de entrada necessários, que no caso do algoritmo genético são a lista de genes, a taxa de cruzamento, a taxa de mutação, o tamanho desejado da população, o número máximo de iterações ou gerações e o número de elementos que vão ser sujeitos à seleção por torneio (linha 16).

```
1  // Construtor do objeto SOAP
2  ServiceReference1.ProductionOperationsSoapClient ws = new
       ServiceReference1.ProductionOperationsSoapClient();
3  // Construtor de uma DataTable
4  DataTable orders = new DataTable();
5  // Armazenamento das ordens na DataTable
6  orders = ws.ReadExcelFile(FilePath, FileExtension);
7  // Construtor do objeto dos resultados do algoritmo
8  List<Gene> AG_Result = new List<Gene>();
9  /* Executa o método algoritmo presente no serviço
10  * Taxa de cruzamento = 50%
11  * Taxa de mutação = 1,5%
12  * Tamanho da população = 100
13  * Numero de gerações = 1000
14  * Tamanho do torneio = 3
15 */
16 AG_Result = ws.ProductionOptimizationAlgorithm(orders, 0.5,
       0.015, 100, 1000, 3);
```

Figura 5.20: Exemplo do código que invoca o serviço ProductionOptimizationAlgorithm

Na figura 5.21 pode ser vista a resposta XML da execução do serviço *Web* responsável pela aplicação do AG. Na mensagem são devolvidos os genes do indivíduo com o melhor valor de aptidão encontrado, ou seja, a sequência pela qual as ordens de fabrico minimizam a função objetivo dos trabalhos atrasados com pesos atribuídos.

De forma semelhante a aplicação do método das prioridades, a aplicação *Web* descodifica a mensagem XML para mostar os resultados ao utilizador, que podem ser vistos

na figura 5.22.

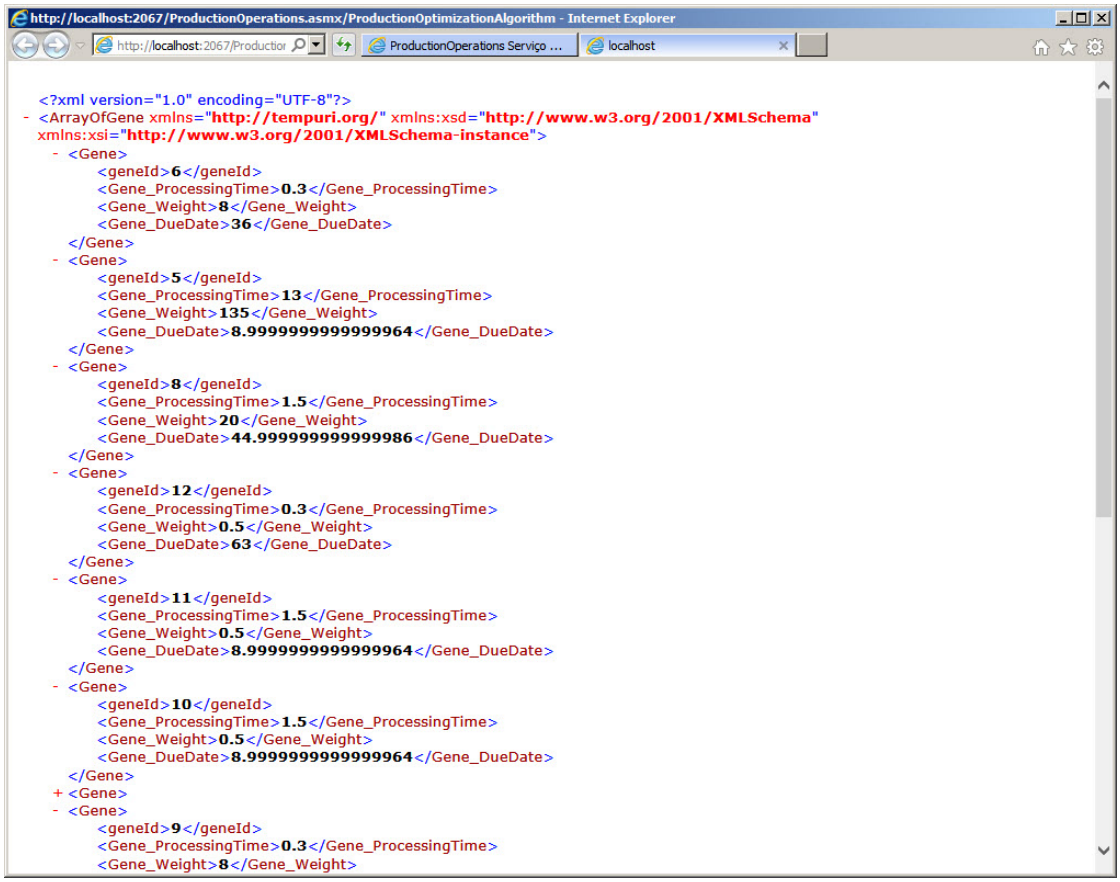


Figura 5.21: Exemplo de uma solução obtida pelo AG

Na página da aplicação podem ser encontradas duas tabelas. A primeira mostra as ordens organizadas segundo as prioridades e o respetivo gráfico do horizonte temporal das ordens que são adquiridas da tabela. Na segunda tabela está representada a sequencia de ordens de produção obtidas da aplicação do algoritmo genético.

De notar que apenas foram consideradas as ordens de produção pertencentes ao grupo A da análise A-B-C da empresa. Embora as ordens restantes entrem no sistema de organização de prioridades, estas não são processadas pelo algoritmo genético que antes de iniciar a execução faz a filtragem das ordens de produção introduzidas.

De seguida é apresentada uma demonstração das diferenças dos resultados obtidos através da aplicação do método de otimização.

Como exemplo, considere-se que existem as ordens de produção presentes na tabela 5.1. Para testar o funcionamento do algoritmo são previstos dois casos possíveis: a existência de datas de conclusão diferentes (Data 1) e o caso extremo em que todas as ordens possuem a mesma data em que devem estar concluídas (Data 2). As datas no exemplo foram atribuídas com vista a um horizonte temporal de cerca de uma semana, de acordo com o que é praticado pela empresa em questão.

Os seguintes resultados foram obtidos para testes efetuados no dia 27 de Junho.

Para o primeiro caso foram obtidos os resultados de $6059,6$ no caso da sequencia

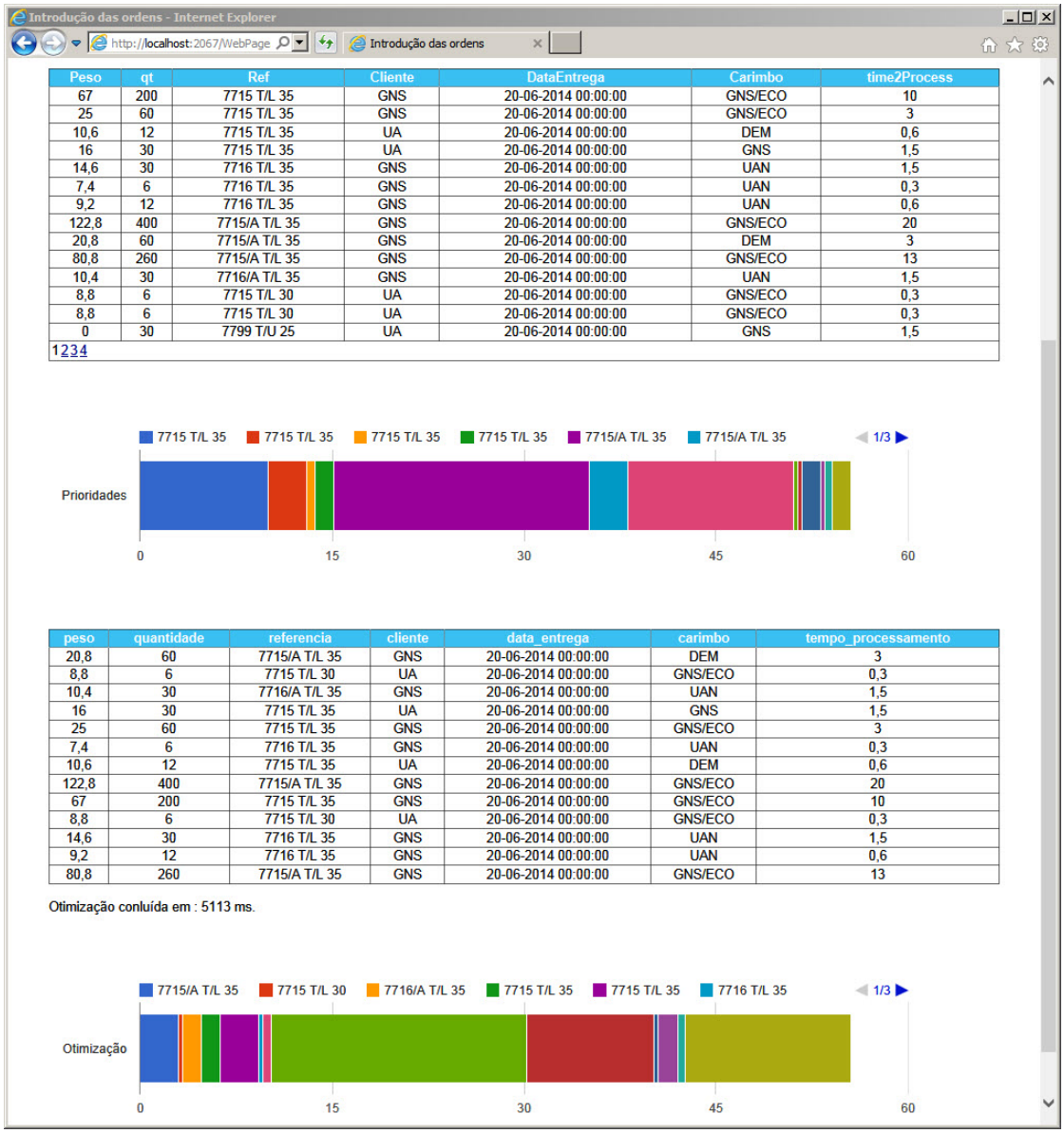


Figura 5.22: Aplicação *Web* dos resultados do AG

Tabela 5.1: Tabela de exemplos de ordens de produção

Quantidade	Referência	Data (Caso 1)	Data (Caso 2)
200	7715 T/L 35	03/07/2014	03/07/2014
60	7715 T/L 35	04/07/2014	03/07/2014
30	7715 T/L 35	01/07/2014	03/07/2014
30	7716 T/L 35	02/07/2014	03/07/2014
06	7716 T/L 35	02/07/2014	03/07/2014
12	7716 T/L 35	04/07/2014	03/07/2014
400	7715/A T/L 35	30/06/2014	03/07/2014
60	7715/A T/L 35	30/06/2014	03/07/2014
260	7715/A T/L 35	05/07/2014	03/07/2014
80	7716/A T/L 35	03/07/2014	03/07/2014

segundo as prioridades e um valor de $4526,74$ para a sequencia obtida pelo algoritmo. Para os dados introduzidos, através do algoritmo foi redução de cerca de $25,3\%$ para a equação do número total de tarefas atrasadas com pesos atribuídos, $\sum w_j \cdot T_j$.

Segundo a otimização, as tarefas passariam a ser sequenciadas segundo a tabela 5.2 , em relação à sequencia da tabela 5.1.

Tabela 5.2: Resultados da otimização para o caso 1

Quantidade	Referência	Data
80	7716/A T/L 35	03/07/2014
30	7716 T/L 35	02/07/2014
60	7715/A T/L 35	30/06/2014
30	7715 T/L 35	01/07/2014
400	7715/A T/L 35	30/06/2014
60	7715 T/L 35	04/07/2014
12	7716 T/L 35	04/07/2014
06	7716 T/L 35	02/07/2014
260	7715/A T/L 35	05/07/2014
200	7715 T/L 35	03/07/2014

A página da aplicação *Web* com os resultados pode ser visualizada na figura 5.23.

Para o segundo caso em que são introduzidas datas comuns para todas as ordens, assiste-se a uma redução de $39,5\%$ segundo o uso do algoritmo genético cujo resultado é de $2092,34$ contra o valor de $3458,6$ obtido através do sequenciamento segundo as prioridades do grupo de produtos.

Segundo os resultados do algoritmo, as ordens de produção da tabela 5.2 poderiam ser sequenciadas segundo a tabela 5.3 para a minimização do total de tarefas atrasadas com pesos atribuídos.

Na figura 5.24 podem ser encontrados os resultados representados através da aplicação *Web* desenvolvida neste trabalho.

Foram ainda feitos mais alguns testes para avaliar o tempo de execução do algoritmo

Tabela 5.3: Resultados da otimização para o caso 2

Quantidade	Referência	Data
60	7715/A T/L 35	03/07/2014
80	7716/A T/L 35	03/07/2014
12	7716 T/L 35	03/07/2014
60	7715 T/L 35	03/07/2014
260	7715/A T/L 35	03/07/2014
400	7715/A T/L 35	03/07/2014
30	7716 T/L 35	03/07/2014
06	7716 T/L 35	03/07/2014
200	7715 T/L 35	03/07/2014
30	7715 T/L 35	03/07/2014

para diferentes conjuntos de tarefas. Os resultados foram obtidos numa máquina equipada com um processador Intel $i7$ a $1,73\ GHz$ e $8,00\ Gb$ de memória RAM e podem ser consultados na tabela 5.4.

Tabela 5.4: Tempos de execução do AG

Dimensão do problema	Tempo de CPU
10 tarefas	2,8 seg
20 tarefas	51 seg
30 tarefas	5 min, 11 seg
50 tarefas	10 min, 55 seg
100 tarefas	18 min, 23 seg

Da análise da tabela verifica-se que o AG desenvolvido é capaz de obter soluções ótimas para problemas pequenos, ente 10 a 20 tarefas e até mesmo 30 tarefas.

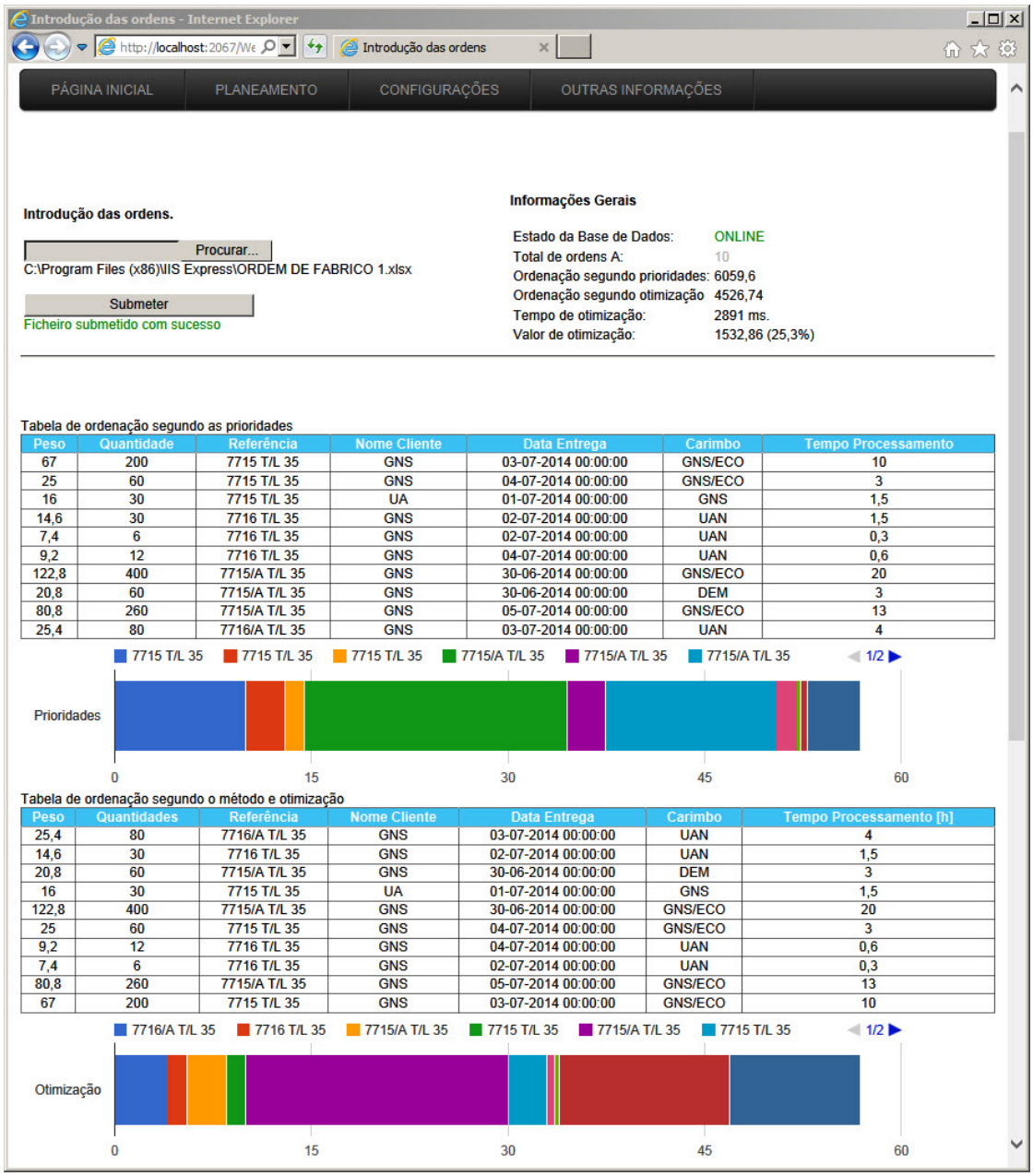


Figura 5.23: Página dos resultados obtidos para o caso 1

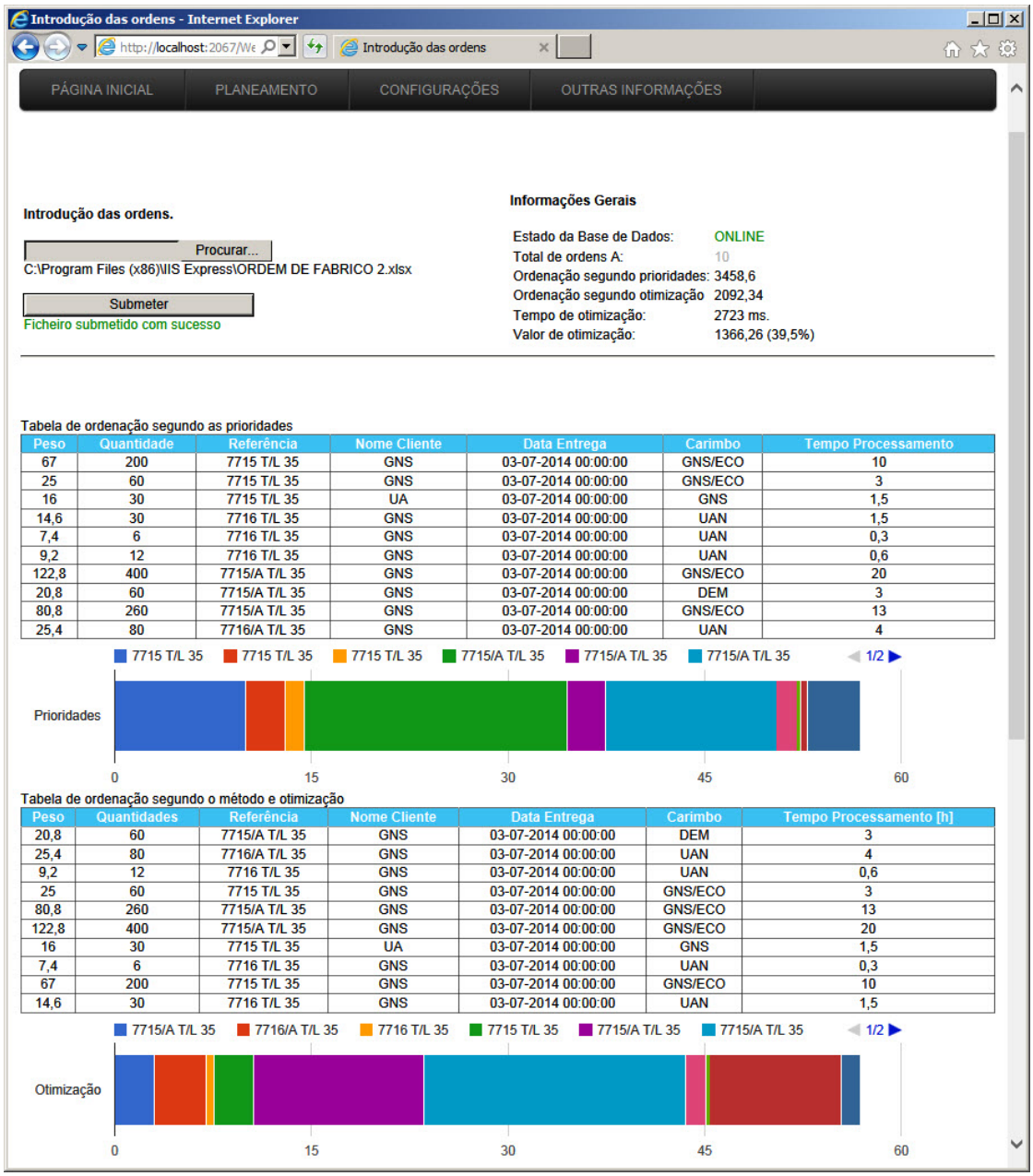


Figura 5.24: Página dos resultados obtidos para o caso 2

Capítulo 6

Considerações finais

6.1 Análise de resultados e conclusões

A maior utilização de sistemas de automação por parte das empresas tem levado à crescente necessidade de soluções que permitam a fácil integração dos sistemas. Isto tem levado as empresas a adotarem paradigmas como a arquitetura SOA através de serviços *Web*.

No presente trabalho foi avaliada a adoção da tecnologia dos serviços *Web* como uma ferramenta para promover a implementação do paradigma da arquitetura orientada a serviços por parte dos sistemas de produção. Foi possível verificar a utilidade dos serviços *Web* como uma solução pertinente na atual mudança dos sistemas de informação na direção da arquitetura orientada a serviços.

Em muitos aspetos, os serviços *Web* são simplesmente uma extensão prática das tecnologias de computação distribuídas que têm estado a ser desenvolvidas ao longo dos últimos anos. Porém, a sua confiança na utilização e funcionamento através do protocolo Internet e standards baseados em protocolos *Web* irão aumentar as suas hipótese de sucesso em fornecer a interoperabilidade sobre diversos fabricantes o que vai permitir às organizações enfrentar os desafios de integração e competitividade do mercado global.

Os serviços desenvolvidos pretenderam fornecer uma solução de integração numa arquitetura orientada a serviços bem como uma solução para o caso de estudo referente a um problema de sequenciamento de ordens de produção de uma empresa.

Numa primeira fase foi desenvolvido um serviço constituído por dois métodos: a leitura do ficheiro e ordenação das ordens segundo prioridades. No caso da leitura do ficheiro submetido pelos serviços de gestão, contendo as ordens de produção, foi criada uma estrutura para armazenar os campos contidos na folha de cálculo de forma a permitir a manipulação dos elementos. Após a leitura de todos os campos é implementado o segundo método que faz a ordenação segundo a lista de prioridades definida numa base de dados.

Estes serviços foram implementados com sucesso através de uma aplicação *Web* desenvolvida para testar e demonstrar os resultados da invocação do serviço de ordenação por prioridades.

Numa segunda fase foi analisado o ambiente produtivo e encontrada uma função objetivo que melhor modelava esse ambiente. Este tema já entra no domínio da gestão onde após alguma análise concluiu-se que o ambiente produtivo poderia ser modelado pela determinação do total de trabalhos em atraso com pesos atribuídos a cada um

dos trabalhos. Uma vez que se trata se um problema de complexidade NP os métodos tradicionais não podem ser aplicado e foi então desenvolvido um algoritmo genético para fornecer uma resposta ao problema.

Os resultados do AG são encorajadores em termos de qualidade da solução assim como tempo de execução. Pode ser concluído que o AG é capaz de encontrar soluções ótima para problemas de tamanhos até 30 tarefas em pouco tempo de processamento. O algoritmo desenvolvido pretende dar um contributo para o avanço da utilização deste tipo de métodos por parte dos equipamentos industriais, uma vez que se baseia numa aplicação prática. O procedimento proposto entra em linha com diversos fatores importantes para a industria moderna nomeadamente a redução de trabalhos em atraso baseado em prioridades estabelecidas pelas empresas. Este é um problema muito estudado na literatura e este trabalho permite mostrar a aplicabilidade do método teórico num problema prático.

Desta forma foi possível verificar a facilidade da implementação dos serviços *Web* criados bem como o seu potencial na crescente procura por interoperabilidade entre os recursos e os sistemas de produção e ainda como a sua independência face ao tipo de linguagem de programação usada.

O trabalho foi desenvolvido do ponto de vista académico apresentado as suas limitações e alguns aspetos a ter em conta, nomeadamente a utilização dos serviços *Web* em simultâneo por várias aplicações cliente e os aspetos de segurança inerentes à utilização da Internet como plataforma de comunicação.

6.2 Futuros desenvolvimentos

Este trabalho debruçou-se sobre a vertente de otimização do escalonamento de ordens de fabrico através da utilização das tecnologias que formam os serviços *Web* com base na lista de ordens lançadas pela empresa, utilizando algoritmos genéticos. Contudo, alguns pontos e vertentes não foram explorados. Neste contexto, são apresentadas e propostas algumas das possibilidades para trabalhos futuros, dando em algum modo continuidade ao trabalho desenvolvido:

- Extensão dos serviços *Web* aos equipamentos de produção;
- Incorporar serviços *Web* que permitem a monitorização em tempo real das operações de produção;
- Estudar a utilização de outros algoritmos para a resolução do problema de otimização do lançamento das ordens de fabrico.

Bibliografia

[1] Armando Walter Colombo and Stamatis Karnouskos. Towards the Factory of the Future: A Service-oriented Cross-layer Infrastructure. In European Telecommunications Standards Institute (ETSI), editor, *ICT Shaping the World: A Scientific View*, volume 65-81. John Wiley and Sons, 2009.

[2] André Figueiredo Quintã. Integração de sistemas de produção. Tese de mestrado, Universidade de Aveiro, 2008.

[3] Diogo Silva Costa. Sistemas de supervisão e controlo integrados. Tese de mestrado, Universidade de Aveiro, 2009.

[4] Vitor Manuel Moreira Martins. Integração de sistemas de informação: perspectivas, normas e abordagens. Tese de mestrado, Universidade do Minho, 2005.

[5] Luís Carlos Figueiredo. Serviços web - supervisão e controle de sistemas de produção. Tese de mestrado, Universidade de Aveiro, 2012.

[6] Commission on Engineering Committee on Visionary Manufacturing Challenges and National Research Council Technical Systems. *Visionary Manufacturing Challenges for 2020*. The National Academies Press, 1998.

[7] A.P. Kalogeras, J.V. Gialelis, C.E. Alexakos, M.J. Georgoudakis, and S.A. Koubias. Vertical integration of enterprise industrial systems utilizing web services. *Industrial Informatics, IEEE Transactions on*, 2(2):120–128, 2006.

[8] F.B. Vernadat. Interoperable enterprise systems: Principles, concepts, and methods. *Annual Reviews in Control*, 31(1):137 – 145, 2007.

[9] Hervé Panetto and Arturo Molina. Enterprise integration and interoperability in manufacturing systems: Trends and issues. *Comput. Ind.*, 59(7):641–646, September 2008.

[10] B. Saenz de Ugarte, A. Artiba, and R. Pellerin. Manufacturing execution system - a literature review. *Production Planning & Control*, 20(6):525–539, 2009.

[11] Sev V. Nagalingam and Grier C.I. Lin. Cim - still the solution for manufacturing industry. *Robotics and Computer-Integrated Manufacturing*, 24(3):332 – 344, 2008.

[12] DIRK BEECKMAN. Cim-osa: computer integrated manufacturing?open system architecture. *International Journal of Computer Integrated Manufacturing*, 2(2):94–105, 1989.

[13] A.W. Scheer. *Business Process Engineering: Reference Models for Industrial Enterprises.* Business process engineering / August-Wilhelm Scheer. Springer, 1998.

[14] Marc M. Lankhorst. Enterprise architecture modelling?the issue of integration. *Advanced Engineering Informatics*, 18(4):205 – 216, 2004. Enterprise Modelling and System Support.

[15] Seung Hak Kuk, II Noh Oh, Hyeon-Soo Kim, Jai-Kyung Lee, and Seong-Whan Park. An e-engineering framework based on service-oriented architecture and agent technologies. In *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, pages 429–434, April 2007.

[16] P. Offermann, M. Hoffmann, and U. Bub. Benefits of soa: Evaluation of an implemented scenario against alternative architectures. In *Enterprise Distributed Object Computing Conference Workshops, 2009. EDOCW 2009. 13th*, pages 352–359, Sept 2009.

[17] E.A. Marks and M. Bell. *Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology.* Wiley, 2006.

[18] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: Service-oriented Architecture Best Practices.* The Coad series. Prentice Hall Professional Technical Reference, 2005.

[19] Organization for the Advancement of Structured Information Standards. *Reference Model for Service Oriented Architecture 1.0.* OASIS, July 2006.

[20] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[21] H. Bohn, A. Bobek, and F. Golatowski. Sirena - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains. In *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, pages 43–43, April 2006.

[22] Dan Driscoll, Antoine Mensch, Alain Regnier, and Toby Nixon. Devices Profile for Web Services Version 1.1, January 2009.

[23] Luciana Moreira Sá De Souza, Patrik Spiess, Dominique Guinard, Moritz Köhler, Stamatis Karnouskos, and Domnic Savio. Socrades: A web service based shop floor integration infrastructure. In *Proceedings of the 1st International Conference on The Internet of Things*, IOT'08, pages 50–67, Berlin, Heidelberg, 2008. Springer-Verlag.

[24] F.Sd.L. Filho, A.L.T.B. da Fonseca, A.J. de Souza, F.A. Couto, R.P.R. dos Santos, and L.F. Guedes. Industrial processes supervision using service oriented architecture. In *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pages 4552–4556, Nov 2006.

[25] G. Veiga, J. N. Pires, and K. Nilsson. Experiments with service-oriented architectures for industrial robotic cells programming. *Robot. Comput.-Integr. Manuf.*, 25(4-5):746–755, August 2009.

[26] William A Estrem. An evaluation framework for deploying web services in the next generation manufacturing enterprise. *Robotics and Computer-Integrated Manufacturing*, 19(6):509 – 519, 2003. Leadership of the Future in Manufacturing.

[27] Eric Newcomer, Hugo Haas, David Orchard, David Booth, Francis McCabe, Christopher Ferris, and Mike Champion. Web services architecture. W3C note, W3C, February 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[28] David Orchard, Hao He, and Hugo Haas. Web services architecture usage scenarios. W3C note, W3C, February 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-scenarios-20040211/.

[29] Punnuluk Phaithoonbuathong, Radmehr Monfared, Thomas Kirkham, Robert Harrison, and Andrew West. Web services-based automation for the control and monitoring of production systems. *International Journal of Computer Integrated Manufacturing*, 23(2):126–145, 2010.

[30] A Girbea, C. Suciu, and F. Sisak. Remote monitoring and control of a flexible manufacturing system through a service oriented architecture subtitle as needed. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1–6, June 2011.

[31] Wolfgang Mahnke, Stefan-Helmut Leitner, and Matthias Damm. *OPC Unified Architecture*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[32] A.J. Alvares, J.L.N. de Souza, E.L.S. Teixeira, and J.C.E. Ferreira. A methodology for web-based manufacturing management and control. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 668–673, 2008.

[33] M. Mathes, C. Stoidner, S. Heinzl, and B. Freisleben. Soap4plc: Web services for programmable logic controllers. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 210–219, Feb 2009.

[34] C. Stoidner and B. Freisleben. Invoking web services from programmable logic controllers. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–5, Sept 2010.

[35] S. Karnouskos, O. Baecker, L.M.S. de Souza, and P. Spiess. Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 293–300, Sept 2007.

[36] João Faria Amorim. Um sistema flexível para o escalonamento de operações industriais. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2009.

[37] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 4rd edition, 2010.

[38] W.J. Stevenson. *Operations Management*. McGraw-Hill higher education. McGraw-Hill Higher Education, 2009.

[39] Maria Leonilde Rocha Varela. *Uma contribuição para o escalonamento da produção baseado em métodos globalmente distribuídos.* Tese de douturamento, Universidade do Minho, 2007.

[40] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd edition, 2001.

[41] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. *Complexity of Machine Scheduling Problems*, volume 1 of *Annals of Discrete Mathematics*, pages 343–362. Elsevier, 1977.

[42] Ümit Bilge, Müjde Kurtulan, and Furkan K?raç. A tabu search algorithm for the single machine total weighted tardiness problem. *European Journal of Operational Research*, 176(3):1423 – 1435, 2007.

[43] M. Fatih Tasgetiren, Yun-Chia Liang, Mehmet Sevkli, and Gunes Gencyilmaz. Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. *International Journal of Production Research*, 44(22):4737–4754, 2006.

[44] A Madureira, D. Falcao, and I Pereira. Ant colony system based approach to single machine scheduling problems: Weighted tardiness scheduling problem. In *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, pages 86–91, Nov 2012.

[45] Andreas C. Nearchou. A hybrid metaheuristic for the single-machine total weighted tardiness problem. *Cybernetics and Systems*, 43(8):651–668, 2012.

[46] Imed Kacem. Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date. *Discrete Applied Mathematics*, 158(9):1035 – 1040, 2010.

[47] Fuh-Der Chou. An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem. *Expert Systems with Applications*, 36(2, Part 2):3857 – 3865, 2009.

[48] Mieczys?aw Wodecki. A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem. *The International Journal of Advanced Manufacturing Technology*, 37(9-10):996–1004, 2008.

[49] Yunqiang Yin, Chin-Chia Wu, Wen-Hsiang Wu, and Shuenn-Ren Cheng. The single-machine total weighted tardiness scheduling problem with position-based learning effects. *Computers & Operations Research*, 39(5):1109 – 1116, 2012.

[50] R. LINDEN. *Algoritmos Genéticos (2a edição).* BRASPORT, 2008.

[51] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, USA, 1998.

[52] Aurora Pozo. Computação evolutiva, 2005.

[53] Jean-Jacques Moreau, Noah Mendelsohn, Henrik Frystyk Nielsen, Martin Gudgin, Anish Karmarkar, Marc Hadley, and Yves Lafon. SOAP version 1.2 part 1: Messaging framework (second edition). W3C recommendation, W3C, April 2007. http://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

[54] Ethan Cerami. *Web Services Essentials.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition, 2002.

[55] James Snell, Doug Tidwell, and Pavel Kulchenko. *Programming Web Services with SOAP.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[56] L. Shklar and R. Rosen. *Web Application Architecture: Principles, Protocols and Practices.* Wiley, 2004.

[57] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (wsdl) 1.1. W3C note, W3C, 2001. http://www.w3.org/TR/wsdl.

[58] Tom Bellwood, Steve Capell, Luc Clement, John Colgrave, Matthew J. Dovey, Daniel Feygin, Andrew Hately, Rob Kochman, Paul Macias, Mirek Novotny, Massimo Paolucci, Claus von Riegen, Tony Rogers, Katia Sycara, Pete Wenzel, and Zhe Wu. UDDI Version 3.0.2 http://uddi.org/pubs/uddi_v3.htm, October 2004.

[59] Anders Hejlsberg, Scott Wiltamuth, and Peter Golde. *C# Language Specification.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.